

LHCb UK Students  
Meeting  
28<sup>th</sup> April 2014



# LHCb Simulation(s)

Disclaimer: It is not a tutorial but will concentrate more on how things are done rather than reliability and performance

May slides are for reference - Will go through them quickly

Gloria Corti, CERN



## ■ Introduction

- Why and how Monte Carlo simulations are used in High Energy Experiments and in LHCb

## ■ Walk through the various aspects covered by MC simulations in the LHCb experiment software

- Generation
- Tracking through the detector
- Detector response
- Spill-over

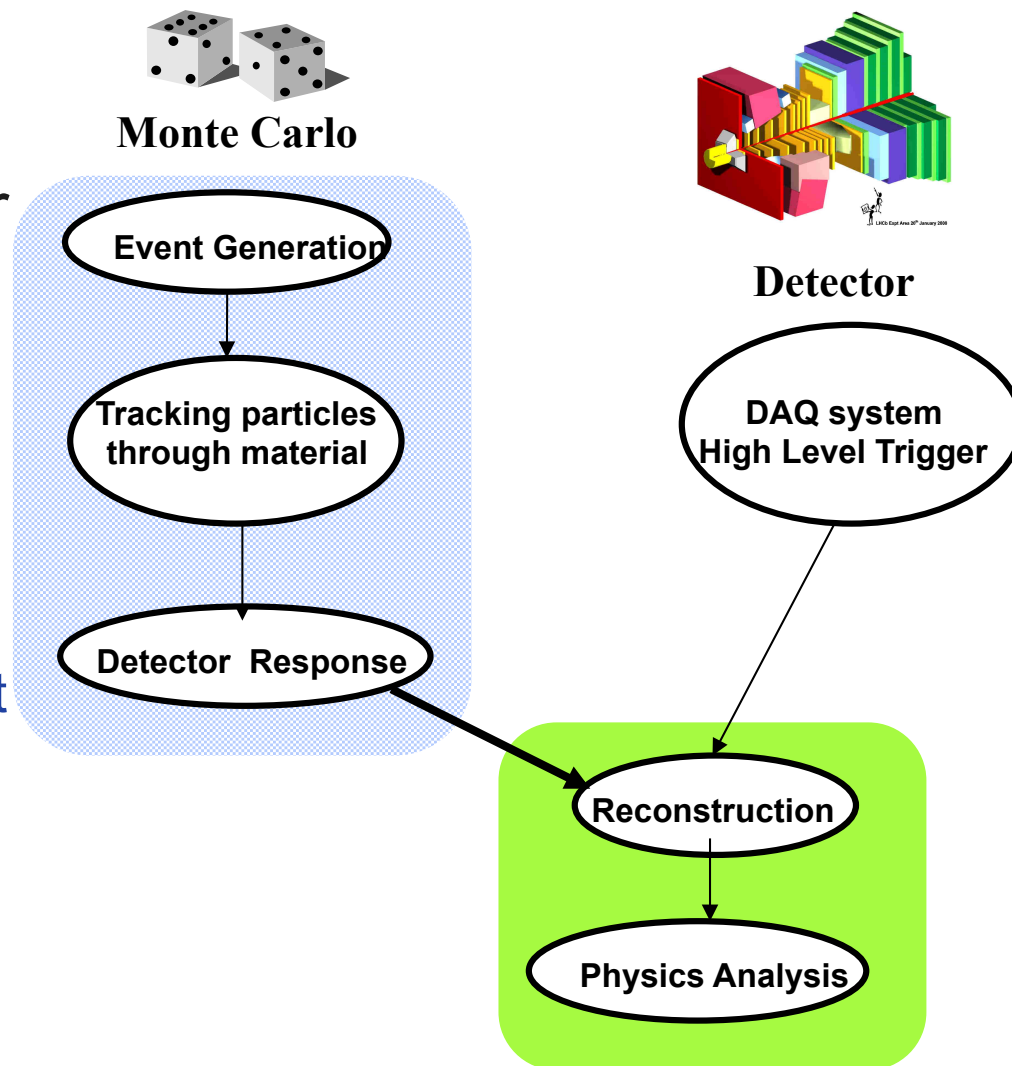
## ■ How does it all fit together in LHCb

- The Gauss and Boole projects and applications
- Aspects of Gauss
  - Steering & Configurable, Output, Monitoring

# Introduction (1/2)

- In High Energy Experiments when elementary particles collides in accelerators (for example)

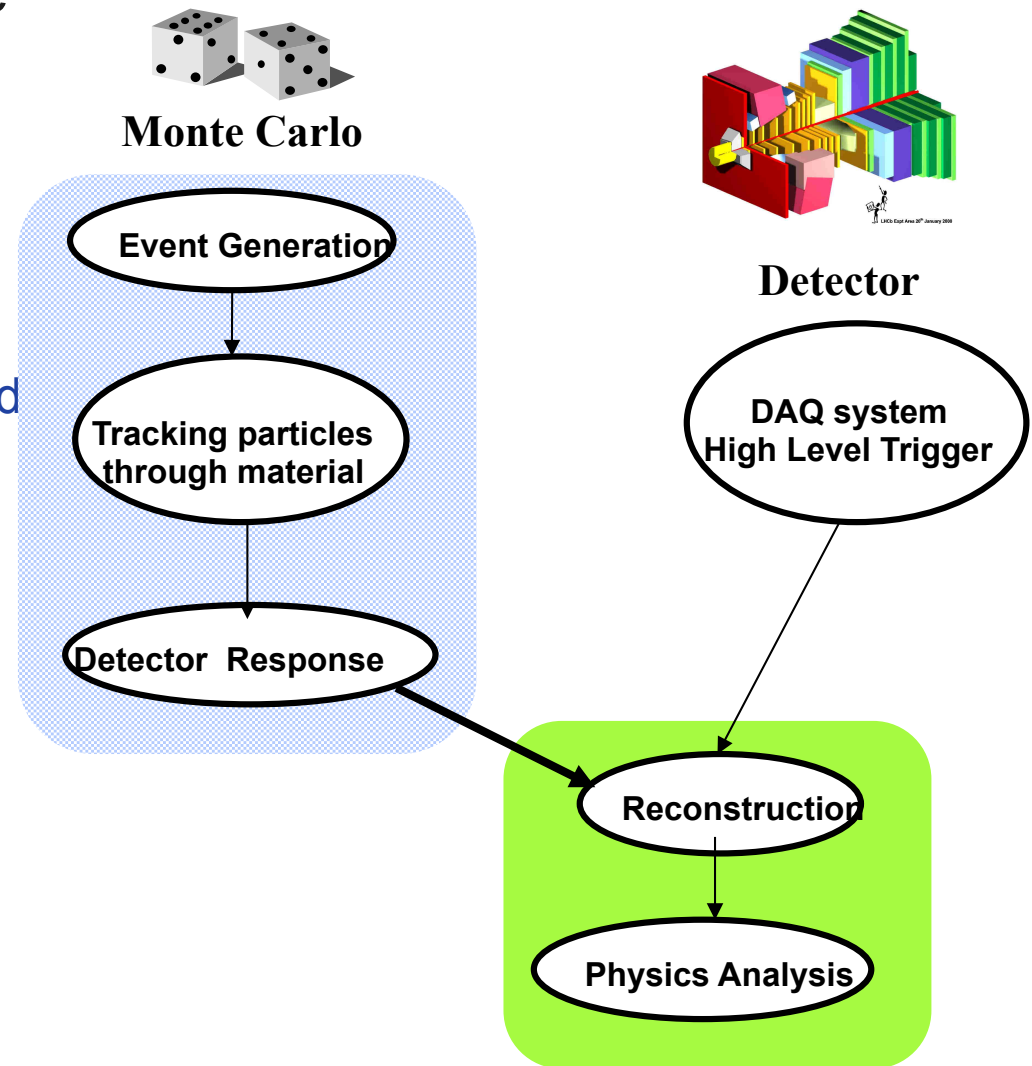
- unstable particles are created, these particles decay quickly.
- it is necessary to reconstruct an “image” of the event through measurements by complex detectors comprising many subdetectors.



# Introduction (2/2)



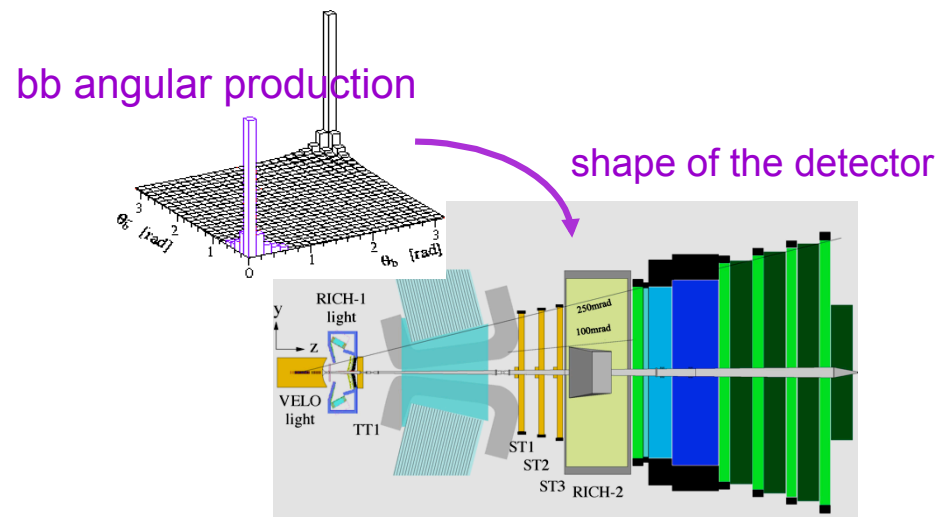
- The Monte Carlo simulation role is to mimic what happens in the spectrometer to understand experimental conditions and performance.
  - Monte Carlo data are processed as real data in Reconstruction and Physics Analysis
    - BUT we know the “truth”
  - Comparing the simulation with what is measured in reality we can interpret the results



# How are MC simulations used ? (1/2)



- Detailed simulations are part of HEP physics
- Simulations are present from the beginning of an experiment
  - Simple estimates needed for making detector design choices
  - Develop trigger, reconstruction and analysis programs
  - Evaluate physics reach
- We build them up over time
  - Adding/removing details as we go along



# How are MC simulations used ? (2/2)



- We use them in many different ways
  - Detector performance studies
  - Providing efficiency, purity values for analysis
  - Looking for unexpected effects, backgrounds
  - When theory is non well known compare to various models and accounts for different detector “acceptance”
- They are also needed to evaluate the radiation environment and its impact on aging maintenance and de-commissioning
- And to evaluate machine interface issues

# Why Monte Carlo simulations? (1/2)



- Generate events in as much detail as possible
  - get average and fluctuations right
  - make random choices,
  - an event with  $n$  particles involves  $O(10n)$  random choices
    - multiple variables: flavour, mass, momentum, spin, production vertex, lifetime,...
  - At LHC (@  $E_{\text{cm}}=7$  TeV) in  $4\pi$ :  $\sim 100$  charged and  $\sim 200$  neutral (+ intermediate stages)
  - → several thousand choices
  
- This applies also to the transport code through the spectrometer and the detectors response
  - want to “track” the particles in the geometrical setup and have them interact with the matter
    - energy loss, multiple scattering, magnetic field
  - want to simulate the detection processes and response of a given detector
    - ionization, scintillation, light
  - the interaction events are stochastic and so is the transport process

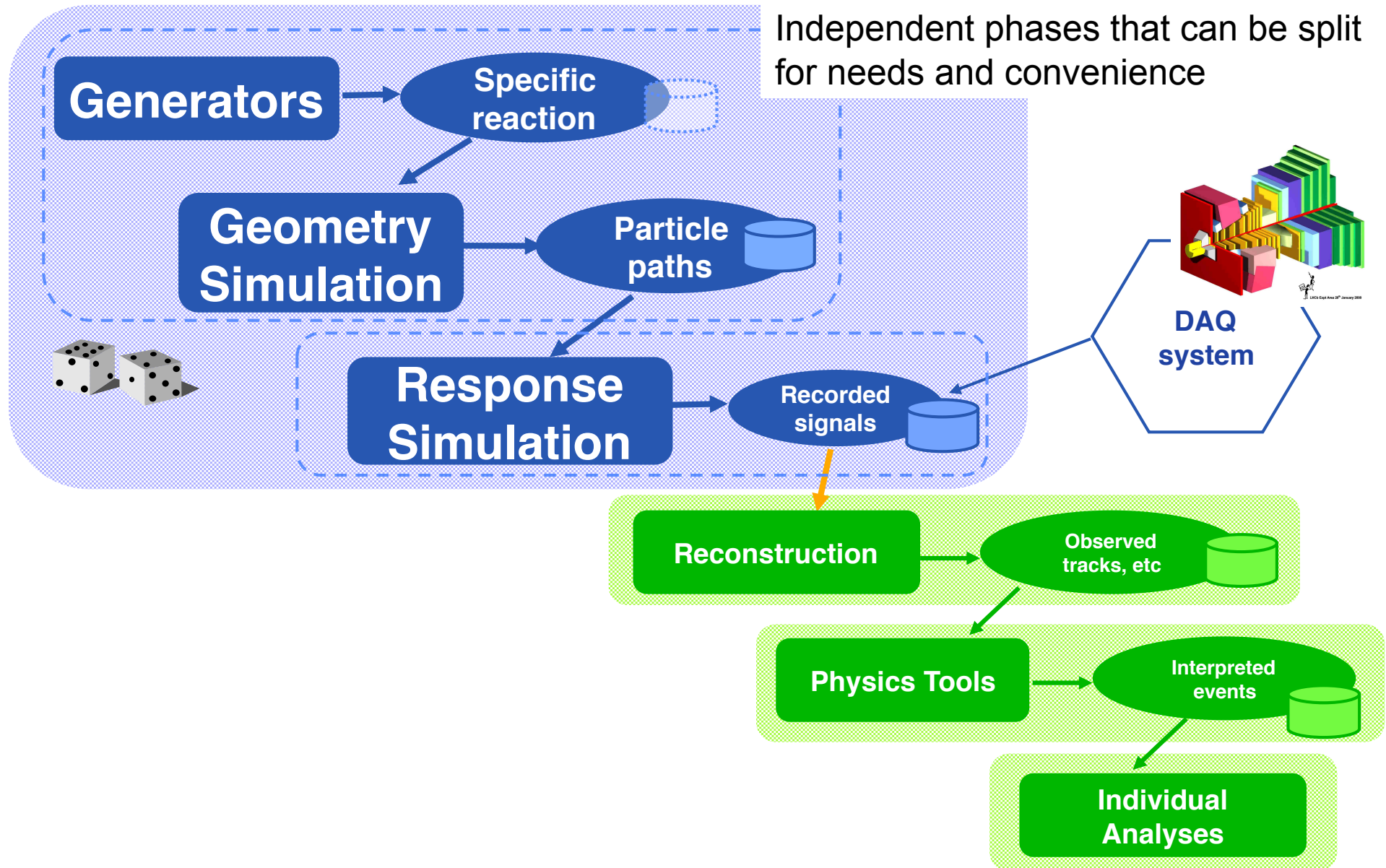
# Why Monte Carlo simulations (2/2)



- A problem well suited for Monte Carlo method simulations
  - computational algorithms relying on repeated random sampling to compute their results
- In fact a Monte Carlo simulation in a High Energy Experiment is a collection of different Monte Carlo, each specialized in a given domain, working together to provide the whole picture
  - it will only do what we tell it do to
  - And contains approximations



# Flow of simulated data and applications





- HEP experiments have their own software frameworks and use external packages developed in the physics community for Generators and for Transport in the detectors
  - Athena (ATLAS), CMSSW (CMS), Gauss (LHCb), VMC (ALICE, CBM@GSI, Minos), bbsim (BaBar), etc.
- Response of the detectors is often in-“house” and requires detectors experts
  - tuned first with test beam data, then with measurements in the experiment
  - in LHCb in a separate application, Boole



- Gauss mimics what happens when collisions (or other events) occur in the LHCb spectrometer. It provides:
  - generation of proton-proton collisions
    - and other type of events (beam-gas, cosmic rays, calibration, ...)
  - decay of particles with special attention to those of b-hadrons
  - handling of beam parameters as luminous regions, pileup (multiple collisions in a beam crossing), spillover (collisions from adjacent beam crossings)
  - tracking of particles in the detector and interactions with the material
  - production of “hits” when particles cross sensitive detectors
- Data produced can be studied directly or in further processing





# Boole – detector digitization

- Boole provides the response of the LHCb detector to the simulated physics events
  - Input are detector hits generated by Gauss
  - From the hits produced by Gauss, generate the digitisations in the raw buffer, as they would come from the online readout system
  - and the MC history, allowing to associate a given digit to the MC particle that generated it.
  - Includes “In-time” hits (signal event) plus any “spillover” hits from previous/next beam crossings enabled in Gauss step
  - Handle also hits from spill-over events if relevant
    - Events produced 25 ns earlier can still produce signal in the detectors
  - Add noise if relevant
    - Electronics noise, long-term radiation background, cross-talk

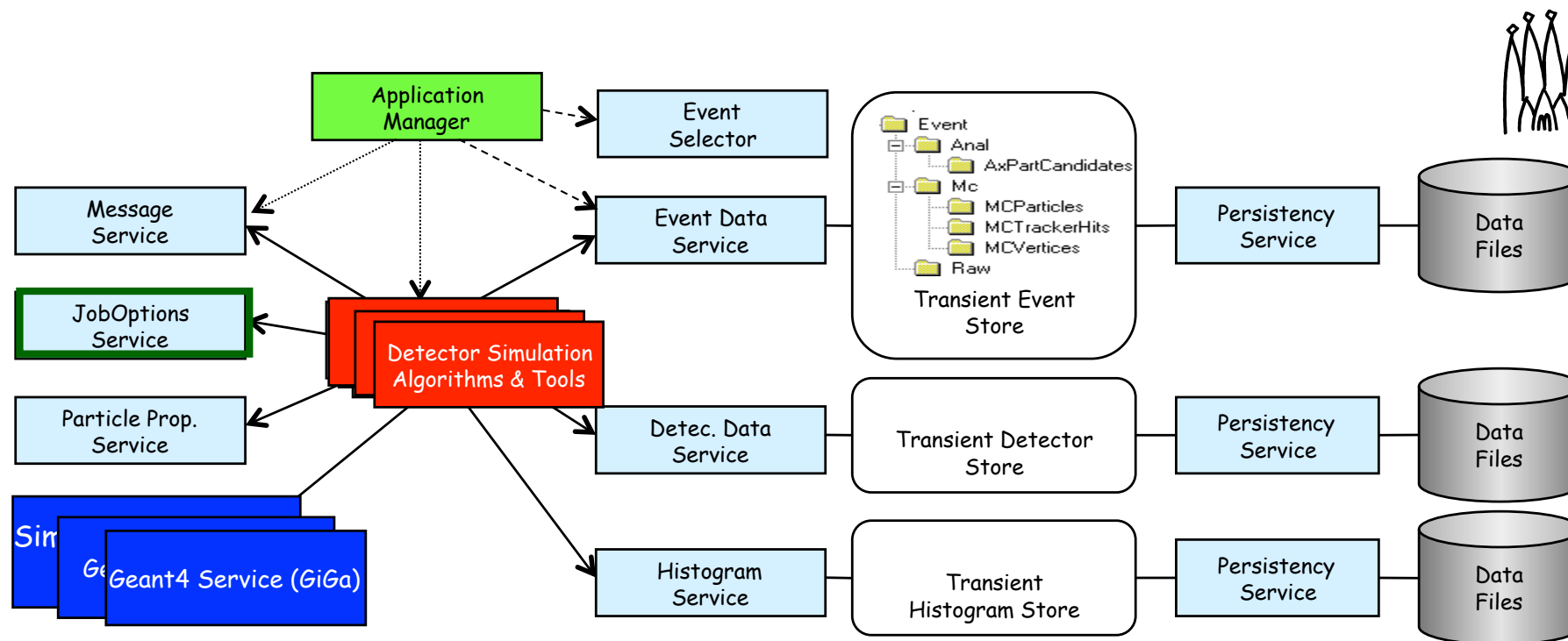
01<sub>10</sub>100111011  
01010001010  
1010<sub>11</sub>0100  
Boole

# Gauss (& Boole): an LHCb application



- Gauss (Boole) is built on top of the Gaudi framework and follow its architectural design
  - same principles and terminology
- Gauss (Boole) as a Gaudi based application is a collection of “User Code” specialized for physics simulation
  - A sequence of algorithms configured via the properties in job configurables
- making use of framework general services
  - JobOptions Service, Message Service, Particle Properties Service, Event Data Service, Histogram Service, Random Number Generator, ...
- LHCb common software
  - LHCb Event model, Detector Description, Magnetic Field Service, ..
- (for Gauss) dedicated simulation software based on external libraries
  - Generator algorithms and tools (Pythia, EvtGen, ...), GiGa (Geant4 Service)

# Gauss (and Boole) are based on GAUDI



- Separation between “data” and “algorithms”
- Separation between “transient” and “persistent” representations of data
- Well defined component “interfaces”
  - Simulation code encapsulated in specific places (*Algorithms, Tools*)

# The Gauss project



- GAUSS is an LHCb software project
  - Contains packages for the Generator phase based on external libraries available from the Physics community
    - Pythia, EvtGen, HepMC, Herwig ...
  - Special interface libraries
    - GENSER (LCG Generator Services): collection of many event Generator libraries
  - Interface Service to Geant4 – GiGa
  - Packages with LHCb detectors simulation code based on GiGa
  - Is based on the LHCb Project, the Gaudi Project and the Geant4 Project



# The Gauss application



- In the Sim/Gauss package
- Gauss() configurable bringing together all the elements
  - Interfaces to event generators
  - Interface to transport code
  - Event model for Generator and MC truth, and Persistency
    - Access to snap-shots of process to understand what happened
  - Histograms, messaging
  - Physicists in the experiments are shielded from Generators and transport code (eg. Geant4) to different degrees

# To run Gauss



- Use `gaudirun.py` as for any other LHCb application with the `Gauss()` configuration you need

By default it will pick the latest version

Define beam parameters: check in `$APPCONFIGOPTS` the different predefined options

```
SetupProject Gauss [v41r2]
gaudirun.py $APPCONFIGOPTS/Gauss/Beam3500GeV_md100_MC11_nu2_50ns.py \
  $GAUSSOPTS/Gauss-DEV.py \
  $DECFILESROOT/options/30000000.py \
  $LBPYTHIAROOT/options/Pythia.py \
  $GAUSSOPTS/Gauss-Job.py
```

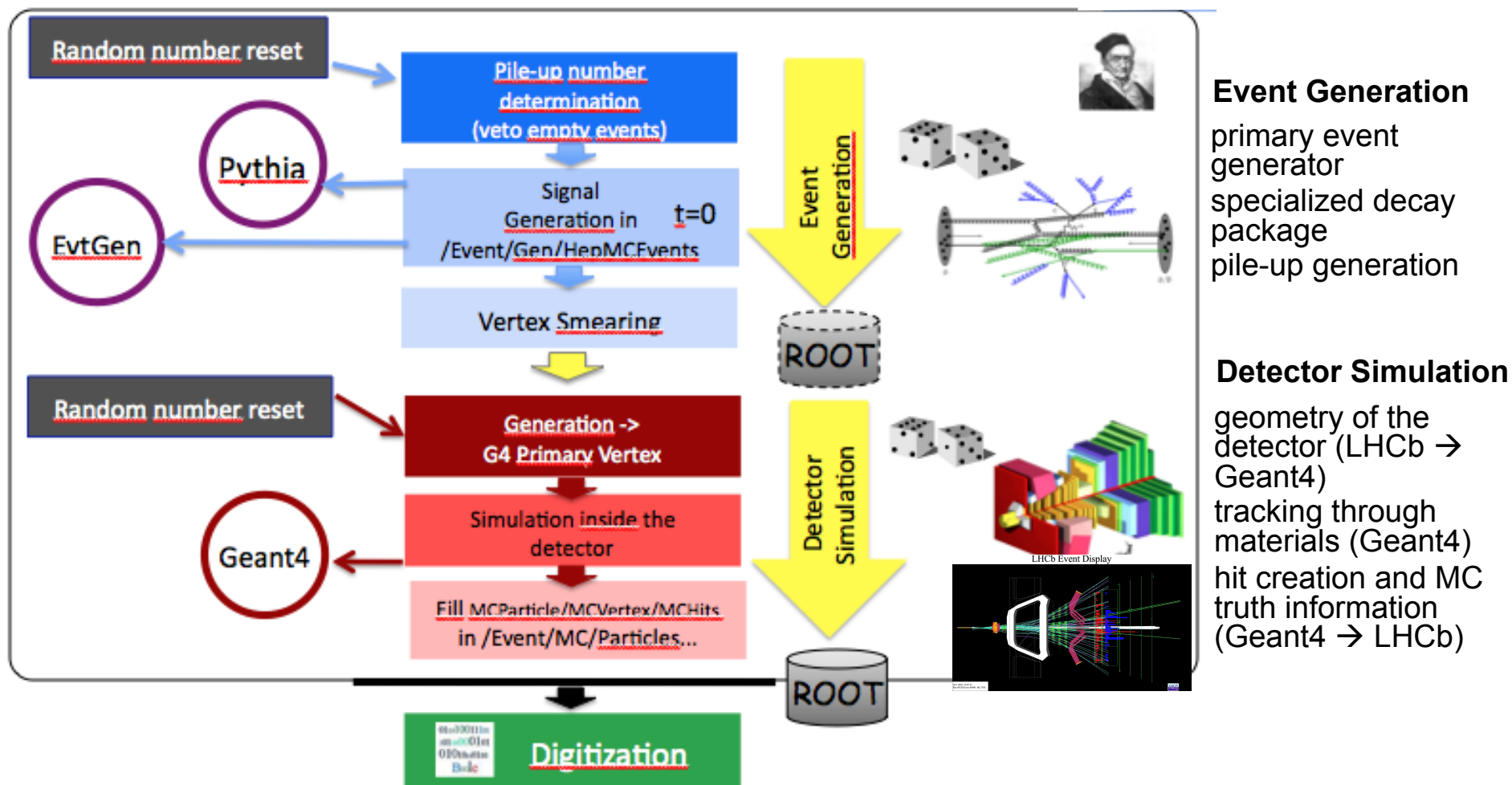
Use latest db tags (can be Overriden in `Gauss-Job.py`)

Specify the type of events to generate and the generator to use. These are the defaults

What is specific for a job, for example how many events (can be copied locally for user's modifications)

# Gauss application sequence

- Two INDEPENDENT phases normally run in sequence
  - but generator phase can and is run by itself

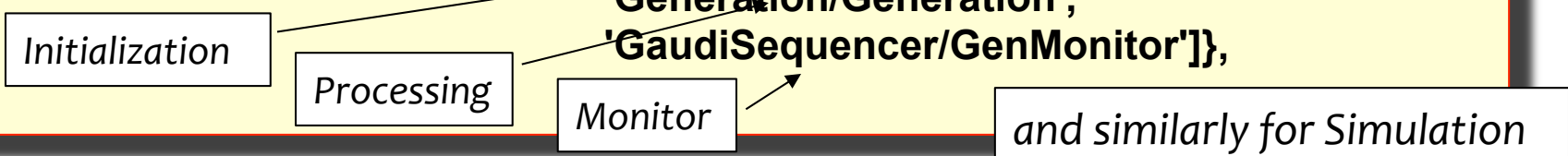


# Phase structure

- Generator and Simulation sequence for each event have its own
  - Initialization – each initializing random numbers and filling their own header
  - Event processing – generating pp collision and decay and transporting through detector
  - Monitor – printing information and filling histograms

```

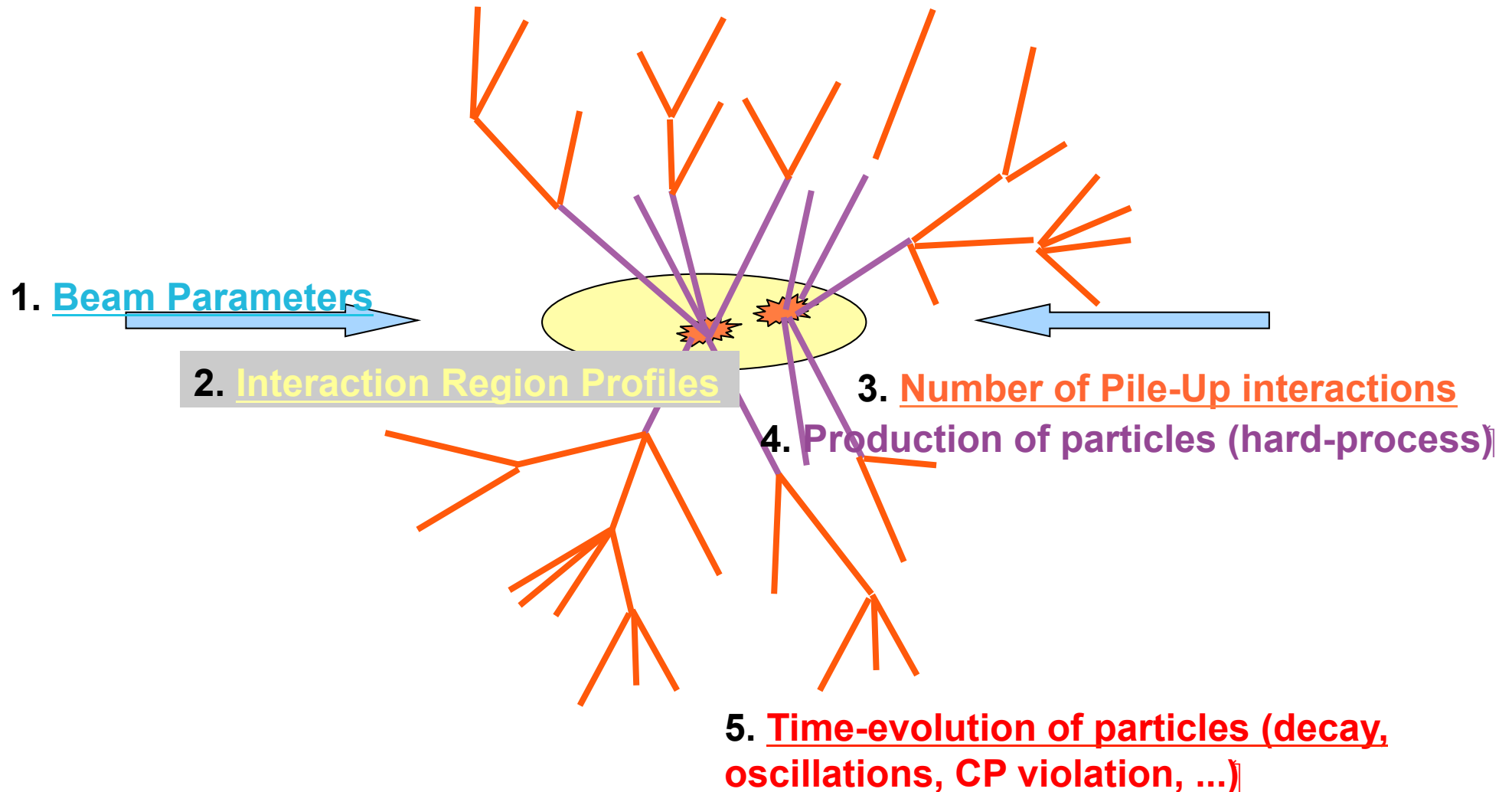
'ApplicationMgr': {'AppName': 'Gauss', ...,
                  'TopAlg': ['GaudiSequencer/GaussSequencer']},
'GaussSequencer': {'Members': ['GaudiSequencer/Generator',
                               'GaudiSequencer/Simulation']},
'Generator': {..., 'Members': ['GaudiSequencer/GeneratorSlotMainSeq']},
'GeneratorSlotMainSeq': {'Members': ['GenInjt/GaussGen',
                                     'Generation/Generation',
                                     'GaudiSequencer/GenMonitor']},
  
```



# Generation part of Gauss



- Takes care of:

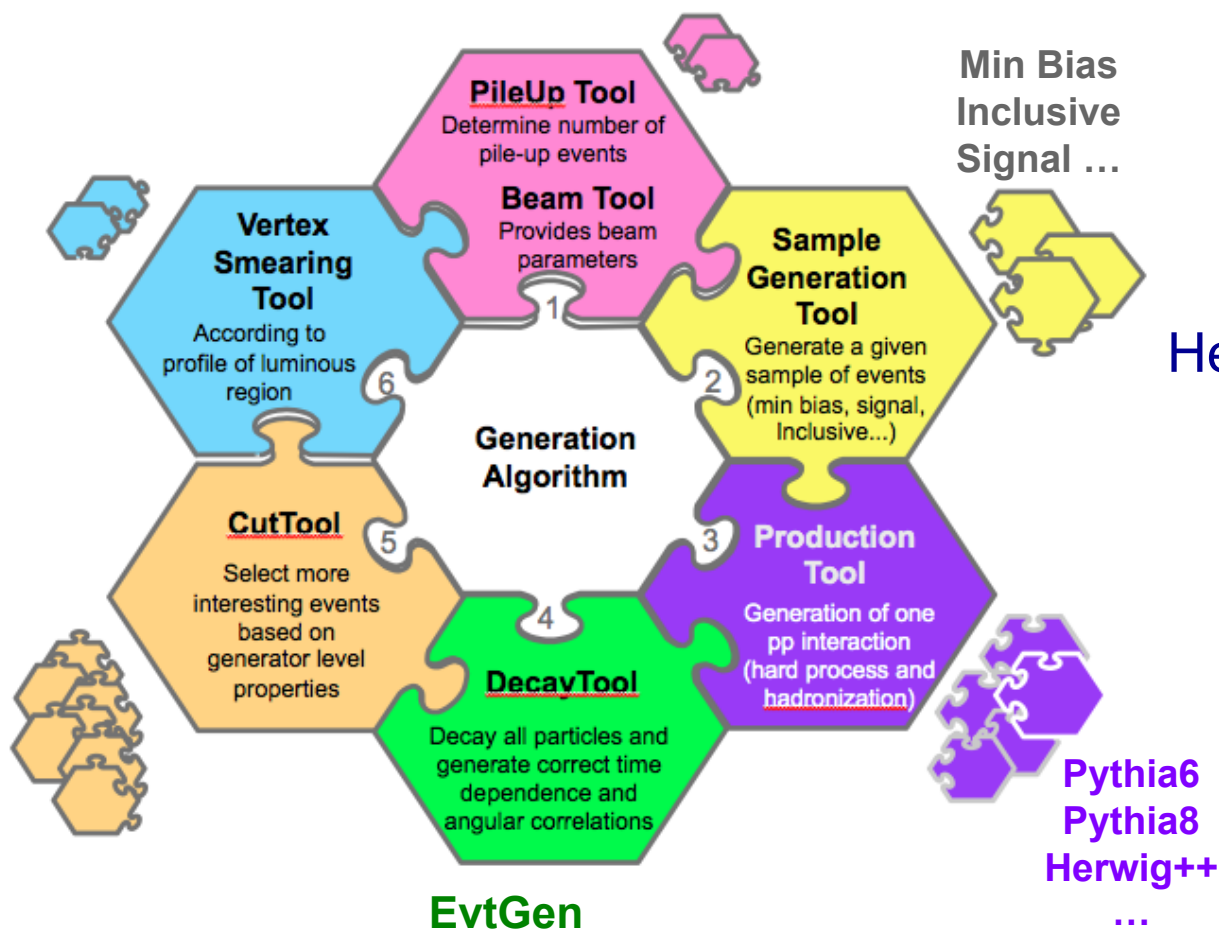


- Many programs available in the physics community to generate primary collisions
  - specialized for hard processes, resonance, decays, parton showers, hadronization, etc.
    - often best at given task, but not always directly usable by experiments
  - general-purpose
    - PYTHIA and PYTHIA8 (T.Sjostrand et al.)
    - HERWIG and HERWIG++ (G.Marchesini et al.)
    - ISAJET (H.Baer et al.)
    - SHERPA (F.Kraus et al.)
  - LCG Generators Service at CERN provides libraries for many of them
    - <http://lcgapp.cern.ch/project/simu/generator/>

- Experiments generally use one generator for massive production and make smaller data sets with others
  - Pythia/Herwig have different hadronisation mechanism (clusters as opposed to strings) for example
  - In LHCb we use Pythia8 as reference generator for the pp interactions at different collision energies to simulate the hard process and the subsequent hadronization and LHAPDF for Parton Density Functions
- and specialized codes when necessary
  - LHCb (ATLAS, Belle, BaBar) uses EvtGen (D.Lange and A.Ryd originally, Warwick now) for b-hadrons decays
  - ATLAS and CMS use MCatNLO (S.Frixione, B.Webber) , AlpGen (M.Mangano et al) , for matrix elements to feed to general purpose – Also LHCb
  - ALICE uses Hijing and EPOS (M.Gyulassy and X.-N. Wang) for Pb-Pb interaction
  - LHCb uses BcVegPy for  $B_c$  production and GenXicc for  $\chi_{cc}$  and  $\chi_{bc}$

# Generator phase

- In LHCb the generator phase is handled by different pieces of code each with a well defined action



HepMC is used as exchange format between these separate objects, in particular the Production (e.g. by Pythia) and the Decay (by EvtGen)





- The most important actions, the functionality of Production and Decay tools, are performed using external libraries, developed outside LHCb and LHCb custom generator libraries
- Gauss is organizing the sequence of actions needed to generate events, calling these external libraries at the right moment, through interfaces.
- The interfaces to the external generators are generic: generators can be exchanged easily only via configurables, for example to use SHERPA instead of PYTHIA.

# Various type of events are produced



- In LHCb, 4 different types of physics events samples can be produced by using different ‘Sample Generation Tools’
  - Minimum bias
    - includes hard QCD processes, single and double diffractive events
  - Inclusive (e.g. generic charm and beauty events)
    - Keep only minimum bias events with a given particle type (or a list)
  - Signal (e.g. b and c signal events)
    - Keep only minimum bias events with a given particle type and force this to decay to a given decay mode
    - It can be done with different methods (SignalRepeatedHadronization, SignalForcedFragmentation, SignalPlain)
  - Special
    - Use special generators or settings to produce very rare processes (e.g. W, B<sub>c</sub>,...)
- Other type of events require different type of algorithms
  - Cosmics and “particle guns” (ie. a given particle with given kinematic) and MIB (Machine Induced Background)

# Minimum Bias (MB) Generation



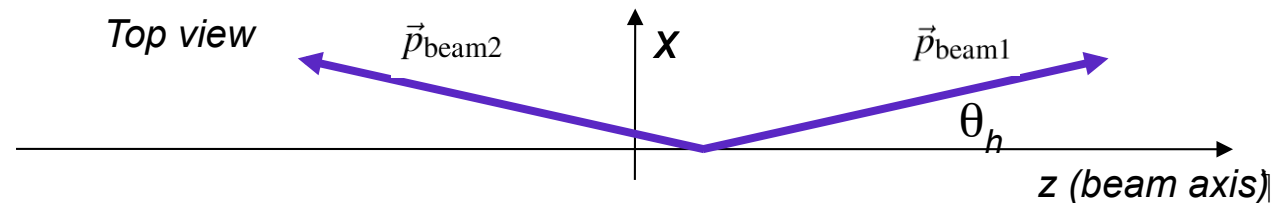
- Most simple generation case is generation of minimum-bias (all what is produced by pp collisions) events.
- Sequence logic is:
  - 1. Generate p beam momentum
  - 2. Determine number N of pile-up interactions
  - 3. Determine space positions of the interactions (PV)
  - 4. Generate N pp collisions
  - 5. Decay all produced particles

- Generates the beam particle ( $p$ ) momenta for each collision.
- Available implementations:
  - CollidingBeams: generates 2 colliding beams with a crossing angle which is smeared by a Gaussian distribution.

$$\vec{p}_{\text{beam1}} = \begin{pmatrix} p \sin \theta_h \\ p \sin \theta_v \\ p \end{pmatrix} \quad \vec{p}_{\text{beam2}} = \begin{pmatrix} p \sin \theta_h \\ p \sin \theta_v \\ -p \end{pmatrix}$$

- where  $\theta_h$  and  $\theta_v$  follow Gaussian distributions with a defined mean value and

$$\sigma = \sqrt{\frac{\epsilon}{\beta^*}}$$



- FixedTarget: generate one single beam,  $\vec{p}_{\text{beam2}} = \vec{0}$

- Determines the number of interactions  $N$  per event.
- Available implementations:
  - FixedLuminosity (default),  $N$  follows a Poisson law of mean value  $n$ , and with  $N \neq 0$ , with

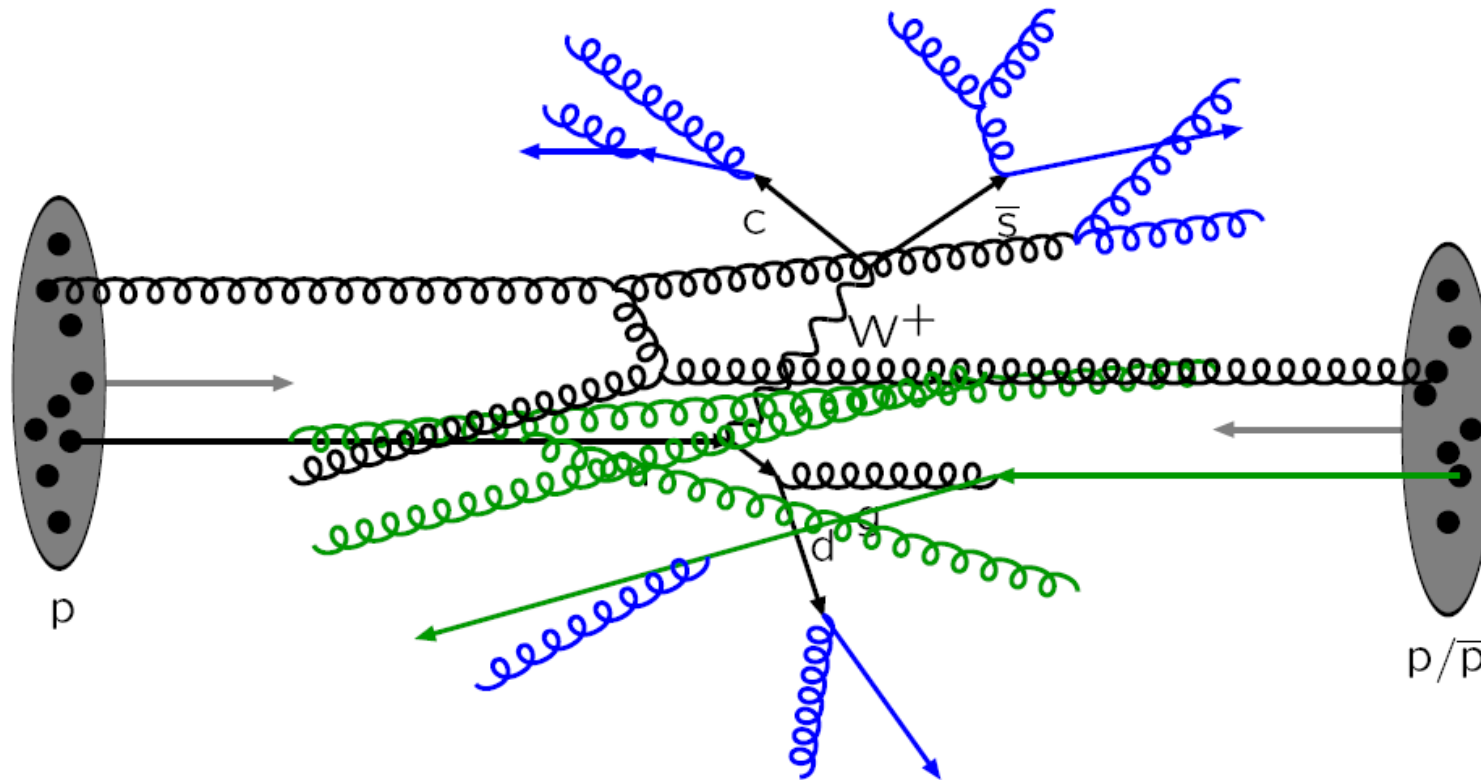
$$v = \frac{\mathcal{L} \sigma_{tot}}{f}$$

- FixedNInteractions:  $N$  is constant.
- FixedLuminosityForRareProcess: used for generation of rare events (see later).  $(N-1)$  follows a Poisson distribution with mean value  $n$ ,
  - The parameters are set exactly the same way than for FixedLuminosity.

# Production Tool



- It is used to generate the pp collisions (hard process, hadronization, ...)



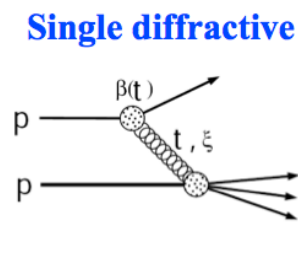
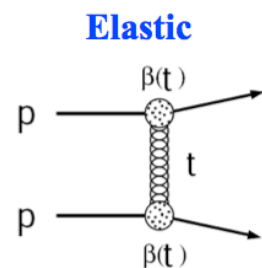
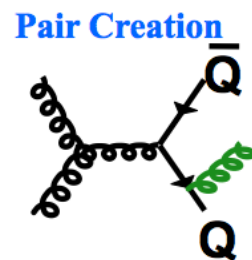
## ■ Available implementations

- PythiaProduction: (being retired) interface to Pythia6
  - Complete FORTRAN generator which contains a lot of different physics processes
  - It can be configured through a large number of switches (common block), described in the Pythia6.4 manual:  
<http://www.thep.lu.se/~torbjorn/pythia/lutp0613man2.pdf>
  - All configuration switches are available to Gauss through Python options
- Pythia8Production: (the new default) C++ Pythia version Pythia8 with the most recent developments
  - Most recent developments only implemented here, e.g. better modeling of diffractive events
  - <http://home.thep.lu.se/~torbjorn/talks/tutorial81.pdf>

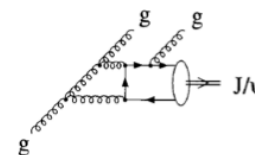
```
myGen.MinimumBias.ProductionTool = "Pythia{8}Production"
```

# Production Tool (Pythia)

- Default options constitute « LHCb tuning », which was done to extrapolate at higher energies charged track multiplicities seen at the UA1 experiment.
- Tuning against LHCb (and other relevant LHC distributions) to define « new tuning »
  - Study of physics parameters involved in Pythia (8 only)
  - Use RIVET and PROFESSOR to compare to reference distributions and scan multiphase parameters
- The activated physics processes are the dominant ones for LHC energies. They define LHCb « minimum bias » out of which all major samples are generated.



**Charmonium production ...**





# Production Tool (others)



- HerwigppProduction
  - General purpose with different production mechanism than Pythia
- HijingProduction
  - For ions interactions
- AlpGenProduction
  - NLO Hard Processes , in LHCb as s input to Pythia6 for now
- PowhegProduction
  - NLO Hard Processes , in LHCb as input to Pythia6 for now
- BcVegPyProduction
  - LHCb in house generator for  $B_c$  production, as input to Pythia6 (and 8 soon)
- GenXiccProduction
  - LHCb in house generator for  $Xi_{bc}$  and  $Xi_{cc}$  production, as input to Pythia6 (and 8 soon)
- SherpaProduction
  - General purpose, with proof of principle in LHCb, also for decays

# Decay Tool (EvtGen)



- It is used to decay all particles, and to generate correct time dependance (CP violation, mixing), correct angular correlations, etc...
- Available implementations:
  - EvtGenDecay: (default) interface to EvtGen
    - Documentation:  
<http://lhcb-release-area.web.cern.ch/LHCb-release-area/DOC/gauss/generator/evtgen.php>

# Decay Tool (EvtGen)



- Particles are identified in EvtGen as strings. The correspondance between PDG Ids and EvtGen particle names is in the database (together with masses, lifetimes, ...):

SetupProject LHCb  
CondDBBrowser DDB &

LHCb name (decay descriptor)

PDGId

Mass

Lifetime

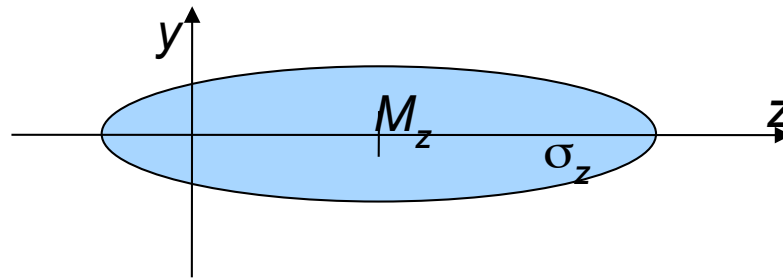
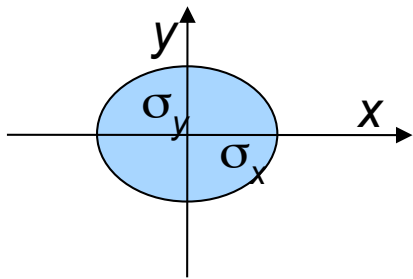
EvtGen name

Name	PDGId	Mass	Lifetime	EvtGen name
eta_c (1S)	441	0.0	2.465214e-023	eta_c
J/psi (1S)	443	0.0	3.09691600	J/psi
chi_c0 (1P)	10441	0.0	3.41475000	chi_c0
chi_c1 (1P)	20443	0.0	3.51066000	chi_c1
h_c (1P)	10443	0.0	3.52593000	h_c
chi_c2 (1P)	445	0.0	3.55620000	chi_c2
eta_c (2S)	100441	0.0	3.63700000	eta_c (2S)
psi (2S)	100443	0.0	3.68609000	psi (2S)
psi (3770)	30443	0.0	3.77292000	psi (3770)
psi (4040)	851	9000443	0.0	psi (4040)
psi (4160)	853	9010443	0.0	psi (4160)
psi (4415)	856	9020443	0.0	psi (4415)
X_1 (3872)	1016	9920443	0.0	X_1 (3872)
X_2 (3872)	1017	9910445	0.0	X_2 (3872)
Z (4430)+	1018	9042413	1.0	Z (4430)+
Z (4430)-	1019	-9042413	-1.0	Z (4430)-
eta_b (1S)	386	551	0.0	eta_b
eta_b (2S)	830	100551	0.0	eta_b (2S)
eta_b (3S)	843	200551	0.0	eta_b (3S)
eta_b2 (1D)	613	10555	0.0	eta_b2 (1D)
eta_b2 (2D)	836	110555	0.0	eta_b2 (2D)
Upsilon (1S)	387	553	0.0	Upsilon

# Vertex Smearing Tool



- Generates profiles of luminous region (position and size).
- Available implementations:
  - BeamSpotSmearVertex: The primary vertex position follows Gaussian distributions in  $(x,y,z)$  and has a fixed time  $t$ .



- FlatZSmearVertex: The primary vertex position follows Gaussian distributions in  $(x,y)$ , a flat distribution in  $z$  and has a fixed time  $t$ .
- ....

- Particle guns (single particle sent to detector)
  - A lot of possibilities already defined in the Gen/DecFiles package (event types starting with “5”)

```
gaudirun.py ... $DECFILESROOT/options/53200010.opts \  
$LBPGUNSROOT/options/PGuns.py ...
```

```
Gauss().Production = “PGUN”
```

- Algorithms/tools in package Gen/LbPGuns:
  - With Fixed Momentum (set px, py, pz)
  - With Gaussian angular spread
  - In a given momentum range
  - Other possibilities can easily be implemented (deriving from base algorithm)

- Accept or reject an event based on generator level quantities
  - Particularly useful when selection signal or background samples to enrich the relevant statistics

- Available implementations:

- LHCbAcceptance: cut on signal direction

$$0 \leq \theta_{signal} \leq 400mrad$$

- DaughtersInLHCb: cut on direction of decay products of signal particle

$$0 \leq \theta_{charged} \leq 400mrad, \quad 10 \leq \theta_{charged} \leq 400mrad,$$

- ...

- Also possible to implement a LoKi based generator level cut

- In the .sim file, generator level events are stored in HepMC format:
  - Documentation: <http://lcgapp.cern.ch/project/simu/HepMC/HepMC203/html/>
- In the TES, events are stored in:
  - Gen/HepMCEvents
- LHCb Specific:
  - Particles have status code (`HepMC::GenParticle::status()`) which have special meanings:
    - 1 = stable in Pythia (p from Primary Vertex, ...)
    - 2 = decayed/fragmentated by Pythia (quark, ...)
    - 3 = Pythia documentation particle (string, ...)
    - 777, 888 = decayed by EvtGen (all unstable particles)
    - 889 = signal particle
    - 999 = stable in EvtGen (p from B decays, ...)
  - Units are LHCb units (MeV, mm, and ns).
  - `HepMC::GenEvent::signal_process_vertex()` is the decay vertex of the signal particle.
- General Information at the collision level kept in an LHCb class, `GenCollision`
  - Mandelstam and Bjorken-x variables, Process type and flag to indicate collision w signal

# HepMC in persistent files

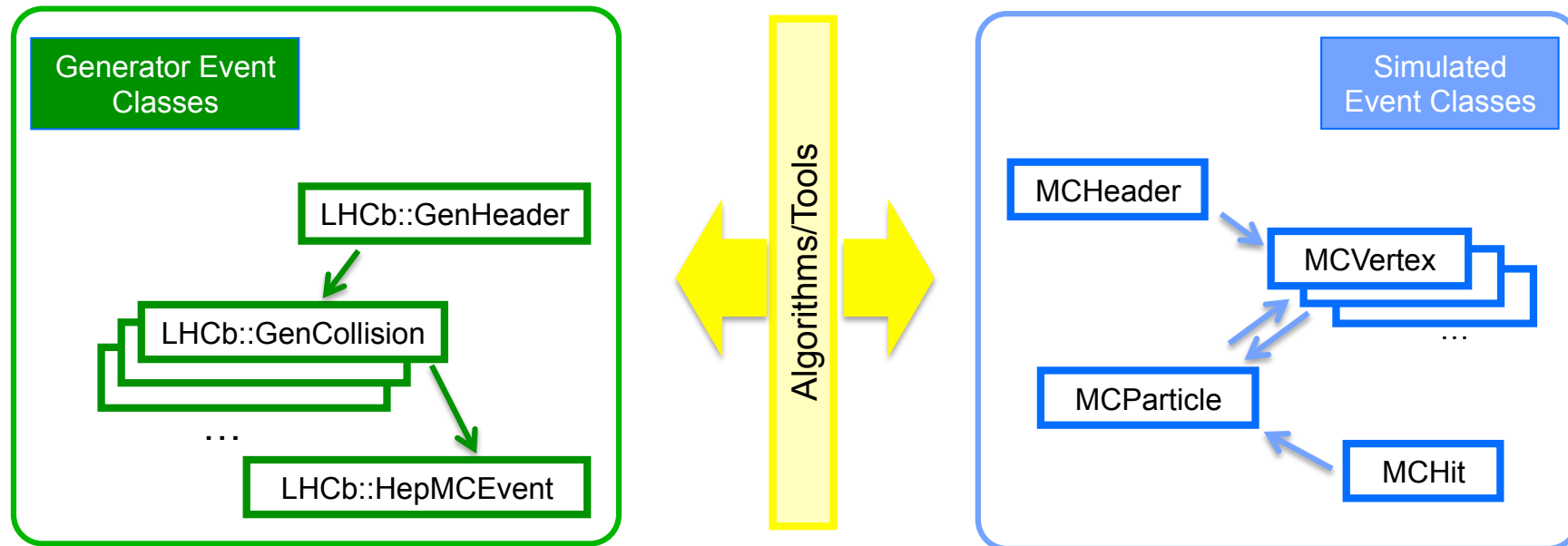


- HepMC was kept in all MC DST files with MC Truth to provide information for analysis of simulated events. But very expensive in term of disk space vs the rest of the LHCb (packed!) event model

	zipped	unzipped
Total	398	1040
HepMC	61	184
MCParticles	46	101
MCVertices	57	107

- Starting from productions made in 2013, HepMC is only kept in generator level productions
  - Heavy quark information needed for flavour tagged analysis and excited b and c hadrons copied to MC Truth
  - will likely be dropped for some of the central productions of this type too
- Users in some cases access directly via ROOT files with only generator level information for their private studies



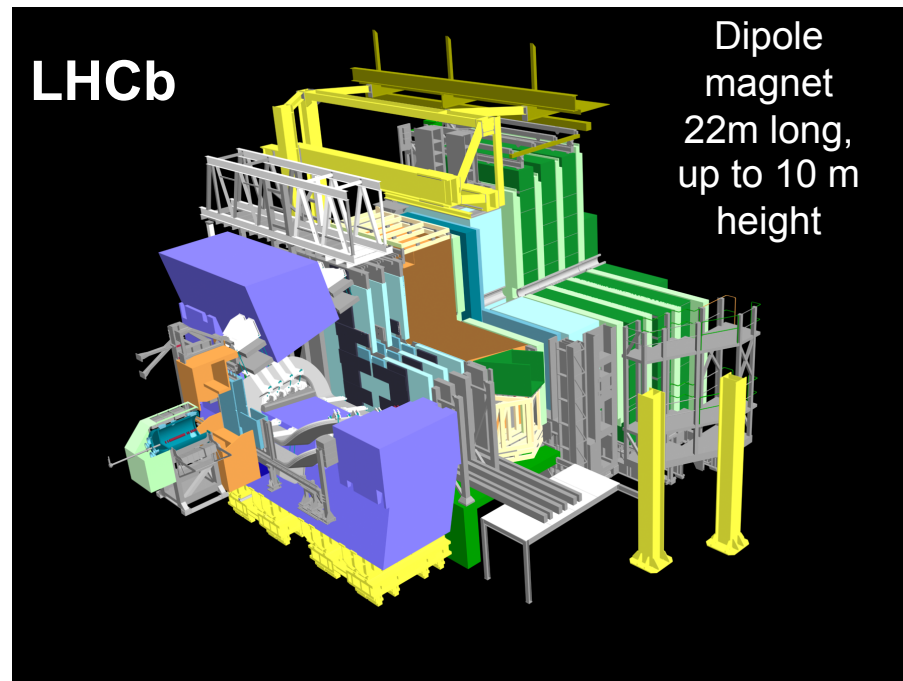


- History of particles traveling through detector in dedicated LHCb event data classes **MCParticles** and **MCVertices** and their relationship, a.k.a. **MC Truth**
- MCParticles/MCVertices contain also the generator level information (decay trees of all hadrons), and only a limited part of the hard process information (heavy quarks, W&Z but not strings, ...).

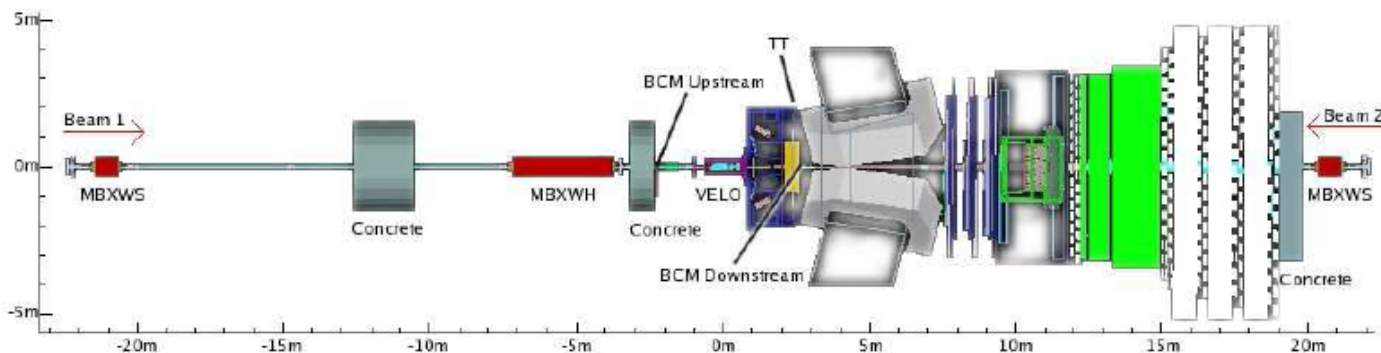
# Detector simulation



Need to describe how particles traverse the experimental setup and what happens to them



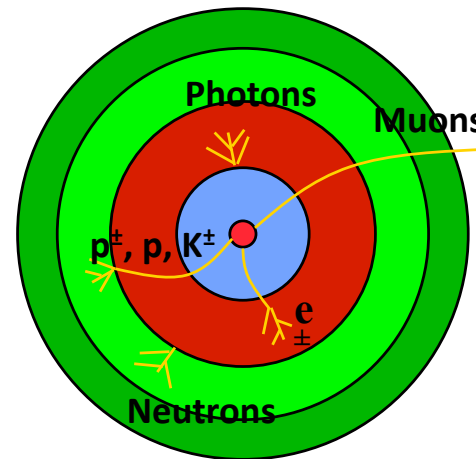
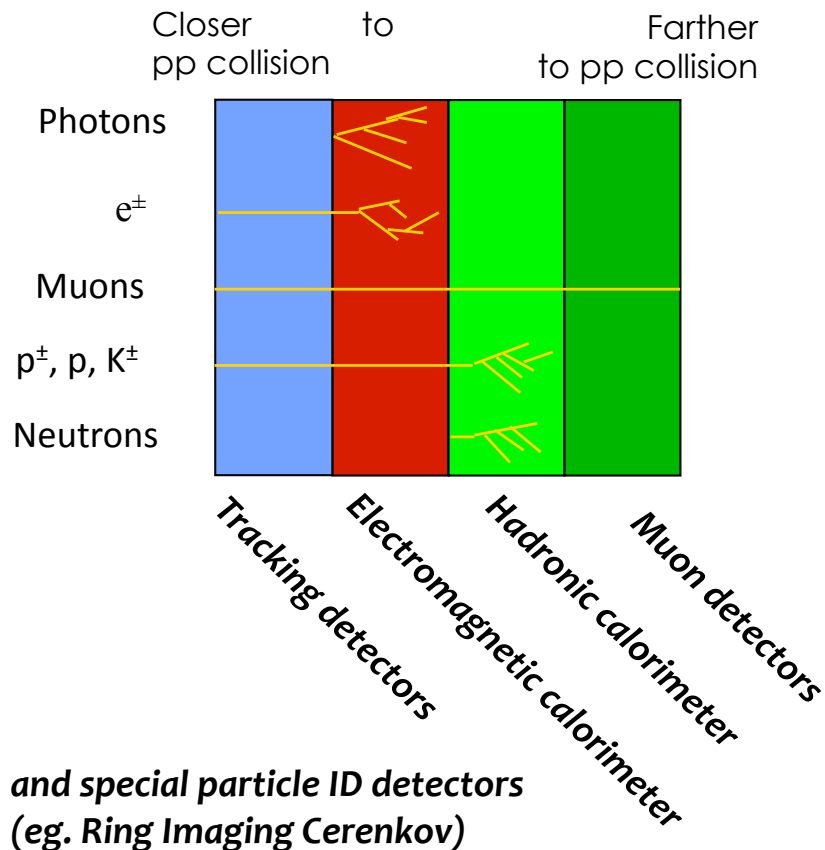
For some studies extend the description to tunnel close by or include the cavern



# Transport through detectors



LHCb and HEP detectors in general are complex and a large variety of physics processes need to be simulated



At LHC particle energies range from few MeV to TeV

What is signal for one detector can be background for another

HEP Experiments use in their framework packages developed in the physics community for transport of particles: GEANT4 and/or FLUKA

# Detector simulation: GEANT4 (1/3)



- LHCb and all the other LHC experiments use GEANT4 for transporting particles in the experimental setup and simulating the physics processes that can occur
  - Navigation in EM fields
  - Physics processes for a variety of particles at different energies
- GEANT4 is a C++ toolkit to track particles through the detector developed in the Physics community
  - GEANT4 International Collaboration ~ 15 year old
  - used in HEP, nuclear physics, heavy ion physics, cosmic ray physics, astrophysics, space science and medical applications
  - GEANT4 is the successor of GEANT3, the world-standard toolkit for HEP detector simulation for the LEP era
  - <http://geant4.web.cern.ch/geant4/>

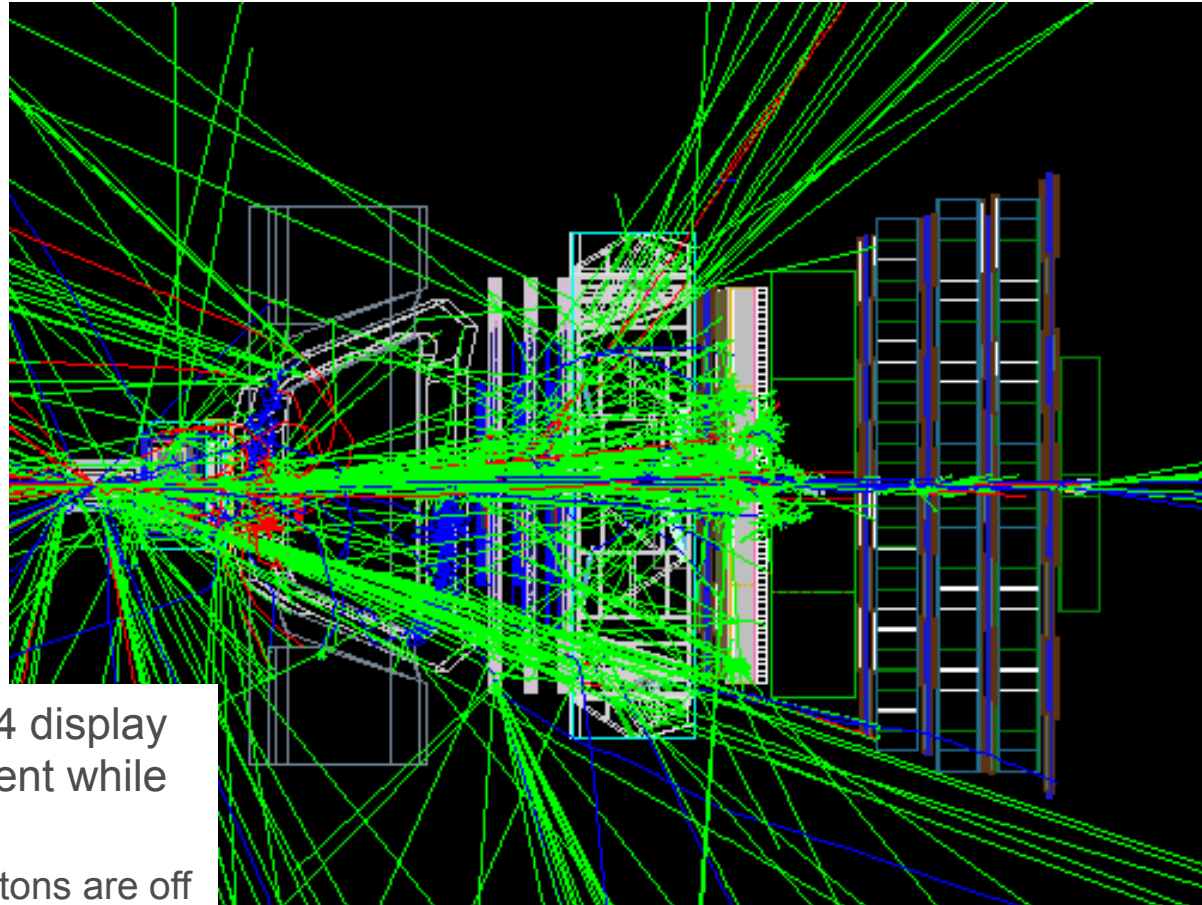


- GEANT4 coverage of physics comes from mixture of theory-driven, parameterized, and empirical formulae. Both cross-sections and models (final state generation) can be combined in arbitrary manners as necessary.
  - Standard and Low energy EM processes, Hadronic processes, Optical photon processes, Decay processes, etc.



- Each particle is moved in steps (few microns to cm)
  - For each step:
    - The material the particle is in is found
    - The energy loss and directional change due to multiple scattering is applied
    - The effects of electric and magnetic fields on charged particles are calculated
    - The particle might generate photons by bremsstrahlung or Cherenkov radiation
    - The particle might decay in flight
    - The particle might collide with a nucleus and generate more particles
    - If the particle enters or leaves a detector it is recorded

Gauss uses a dedicated set Gaudi services, algorithms and tools to interact and communicate with Geant4



Gauss internal Geant4 display of a minimum bias event while being processed.

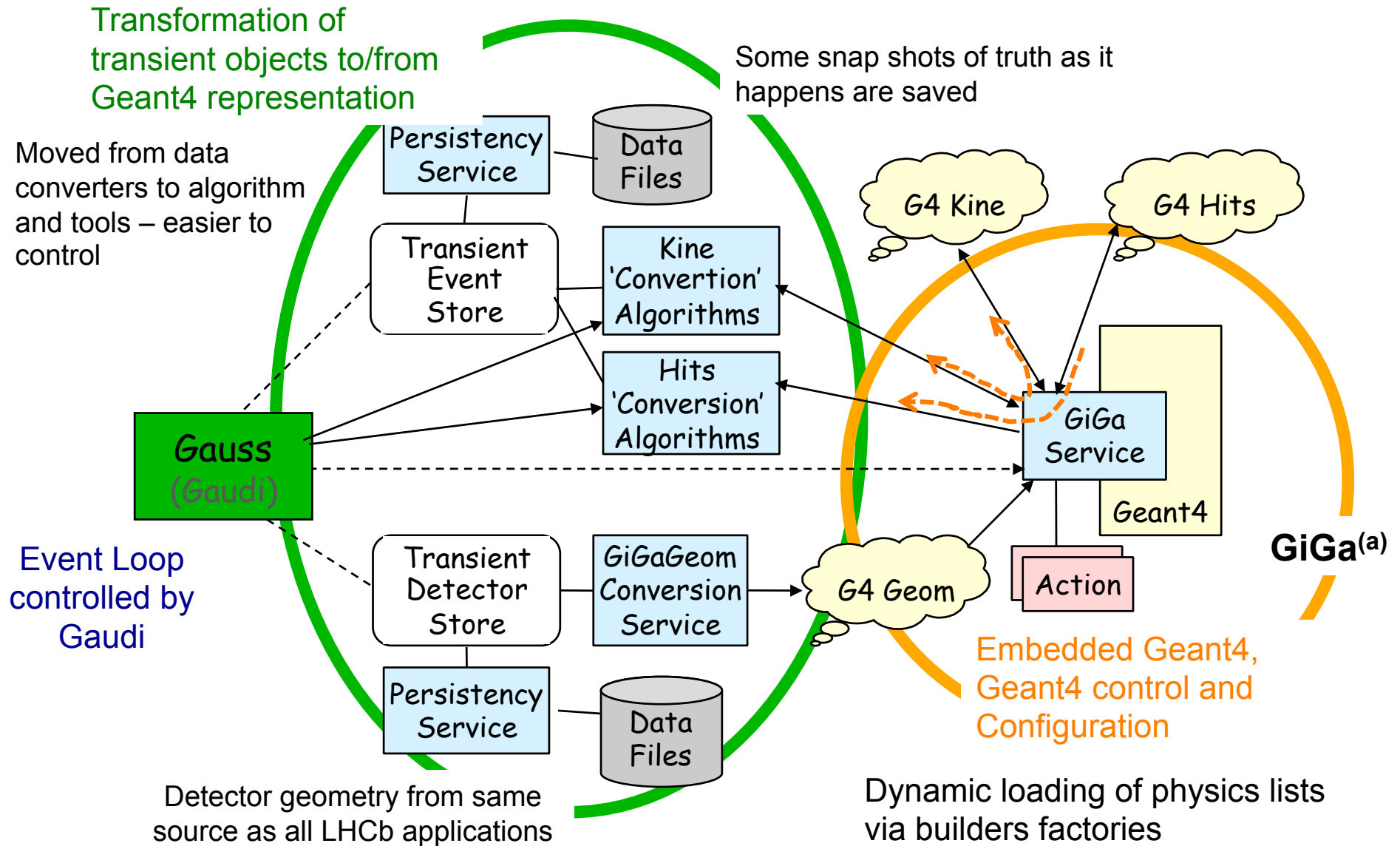
Cerenkov photons are off

In dense material number of particles created and tracked by Geant4 grows quickly and with that the CPU time!

- Gauss uses a dedicated Gaudi service to interact and communicate with Geant4
  - minimizes the couplings to Geant4
- GIGA = GEANT4 Interface for Gaudi Applications or Gaudi Interface to GEANT4 Applications
  - GEANT4 callable and controllable from within GAUDI environment
  - common detector geometry source used by other applications (reconstruction, visualisation)
  - use of Gaudi features as algorithms, tools, services
  - use of common services (ex. RandomNumberSvc, MagneticFieldSvc, DetectorDataSvc, etc.)
  - access to internal Geant4 event loop via GiGaRunManager
  - allows loading external physics lists
  - instantiates (using Abstract Factory pattern) different “actions” (makes them to be pluggable components)



# Encapsulation of Geant4 in Gauss



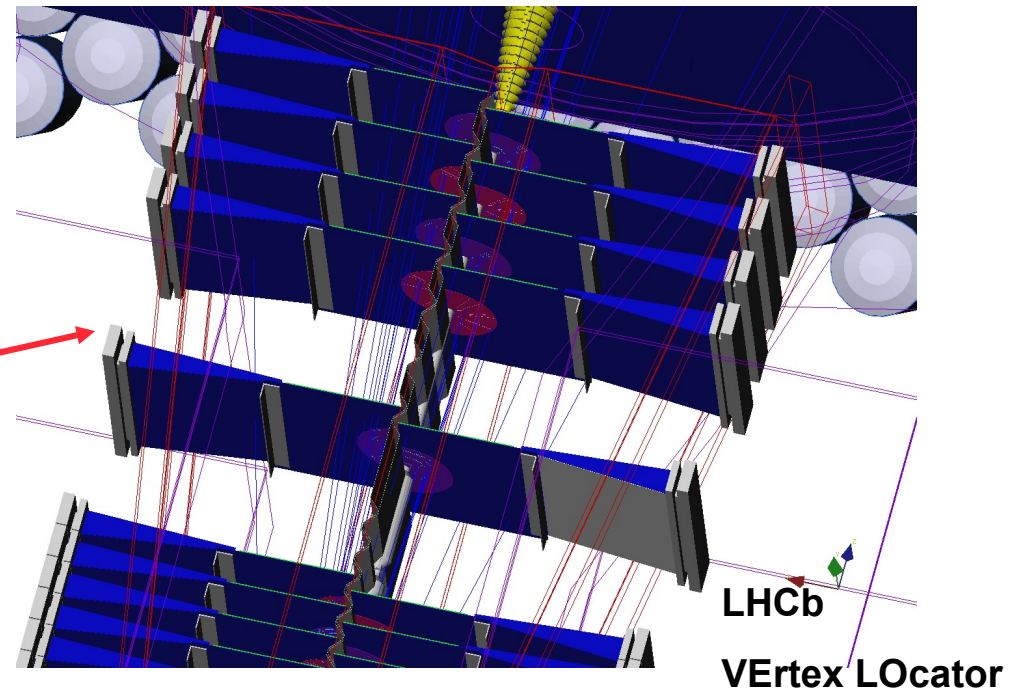
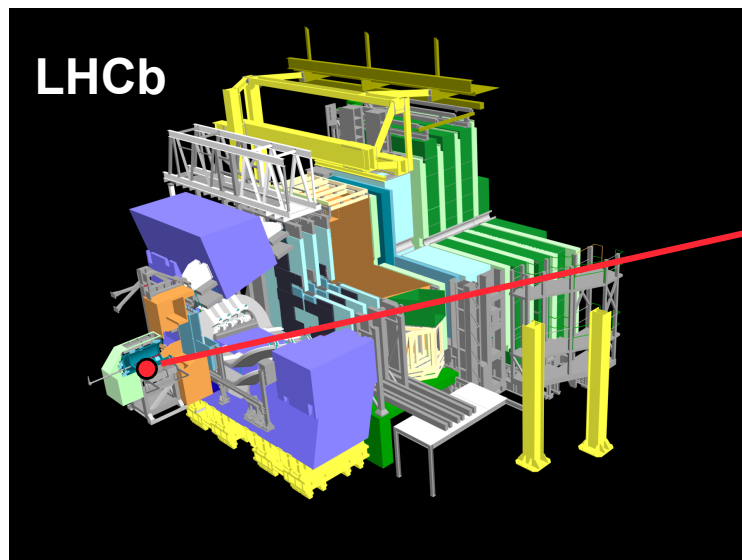
(a) Geant4 Interface for Gaudi Applications (I.Belyaev)

# Geometry modeling



**Need accurate modeling to have the most accurate results but more volumes, more memory, more CPU time**

A wide variety of dimensions in an HEP experiment  
Choose depending on relevance to physics study

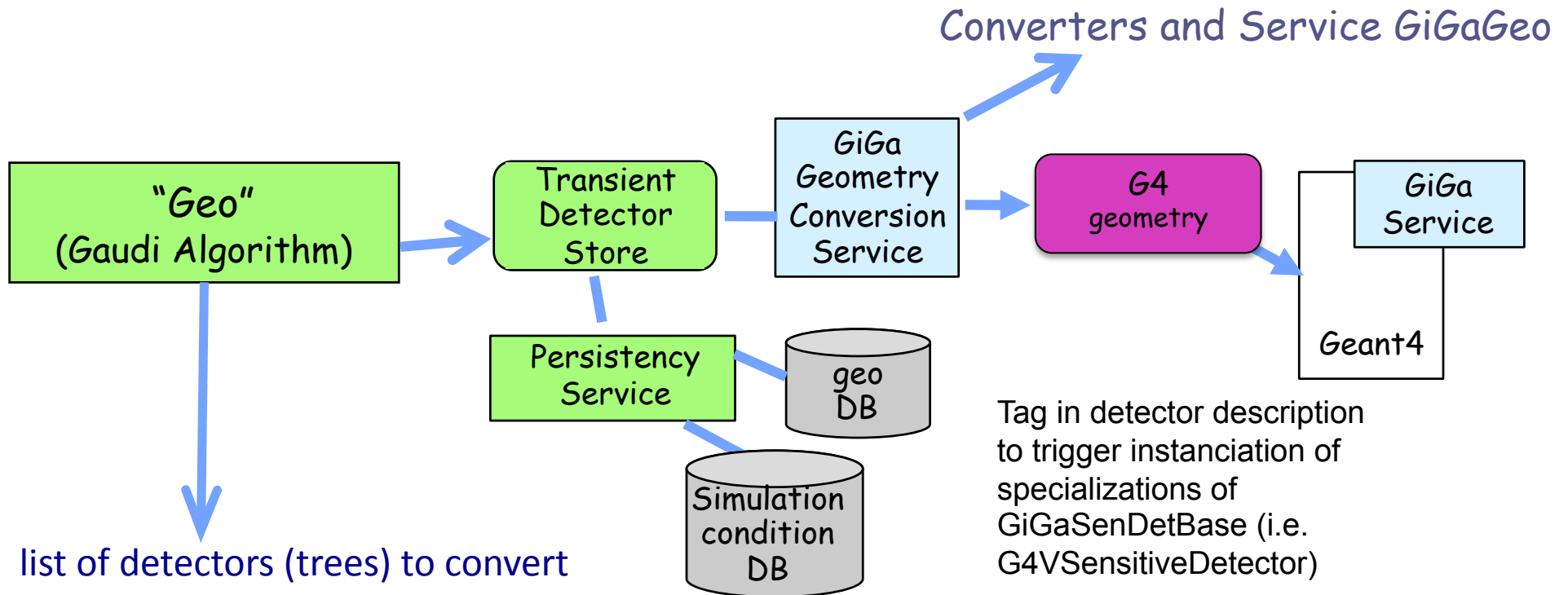


For trackers detailed description of all active and passive components; material budget

# LHCb geometry in the simulation



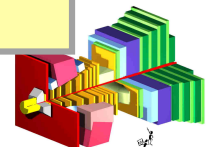
Gauss converts the LHCb geometry to the Geant4 description:



The list controllable for default LHCb detector via a simple Gauss configuration  
Can give different list: switch off some detectors, add some new, test beam or upgrade

```
Gauss().DetectorGeo = { 'VELO': ['Velo', 'PuVeto'], ...  
                        'CALO': ['Spd', 'Prs', 'Ecal', 'Hcal'], ... }
```

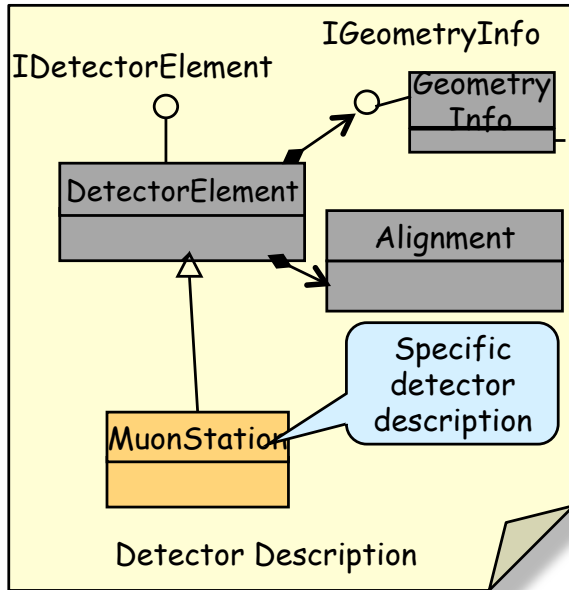
passed to appropriate GiGa component with pathway to find it in Transient Detector Store



# Geometry in the Simulation



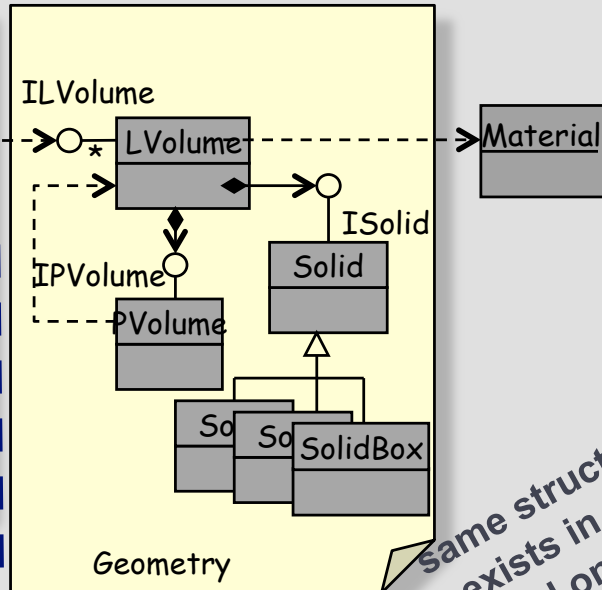
info about misalignment



GiGaDetectorElementCnv

only called when given to GiGa in list to simulate

position as given in LHCb geo DB

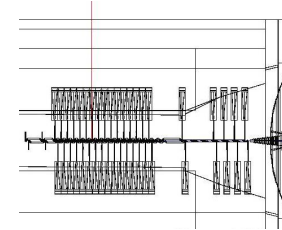


GiGaLVVolumeCnv

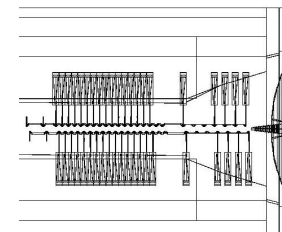
convert to G4 all its geometry tree

same structure exists in G4: mapped one to one

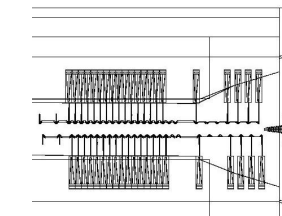
CloseVelo



SemiOpenVelo



OpenVelo



## Limitation:

- Pass all detectors (elements) to mis-align in list to 'Geo' algorithm
- Pass all detectors at same level to what to misalign
- Cannot apply mis-alignment to both parent and their children if information is in condition DB

**Solution:** re-engineering the conversion to G4 geo to take into account the Detector Element structure → as less automatic mapping ensure consistency!

## Need to describe physics processes as accurate as needed by resolution of detectors

Choose appropriate physics models and set cuts: more accuracy → more CPU

- Geant4 has a big variety of processes that can be combined as necessary in Physics Lists (PL):
  - Crucial part of the whole simulation program
  - Library of processes needed are implemented in Geant4
  - Few specific processes have been implemented in Gauss
    - for RICH: photoelectric process (creation of photoelectrons in HPDs), energy loss in the silicon of HPDs
- We use the Geant4 Physics List as baseline to construct list to be used for specific production
  - From a subset of builders available in Gauss as factories to load at run time

- In Gauss a subset of the Geant4 physics builders are available
  - Few variants for hadronic physics:
    - LHEP (our default)
    - QGSP, QGSP\_BERT, QGSP\_BERT\_HP, QGSP\_BERT\_CHIPS
    - FTFP\_BERT
  - All variants of EM standard physics including LHCb tests for MSc and variant of Opt1
    - EmStandard
    - EmStandard\_Opt1 (default up to now, finalizing decision for next MC production)
    - EmStandard\_Opt1\_NoApplyCuts
    - EmStandard\_Opt2, EmStandard\_Opt3
    - EmStandard\_LHCb (+Test)

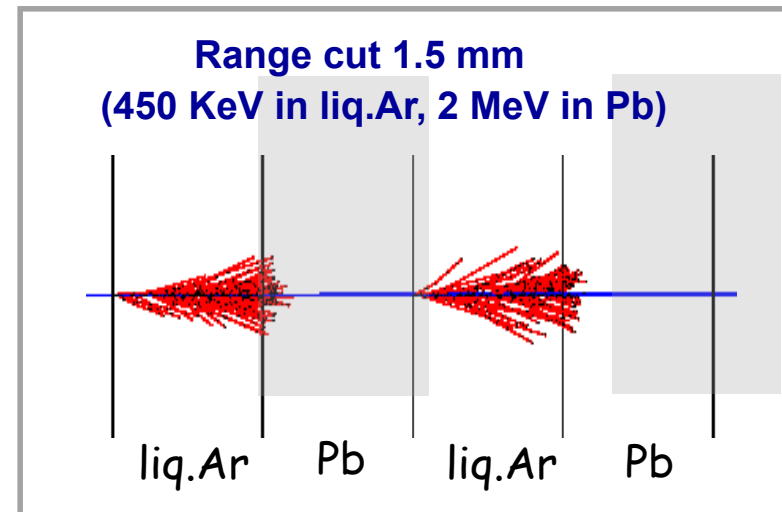
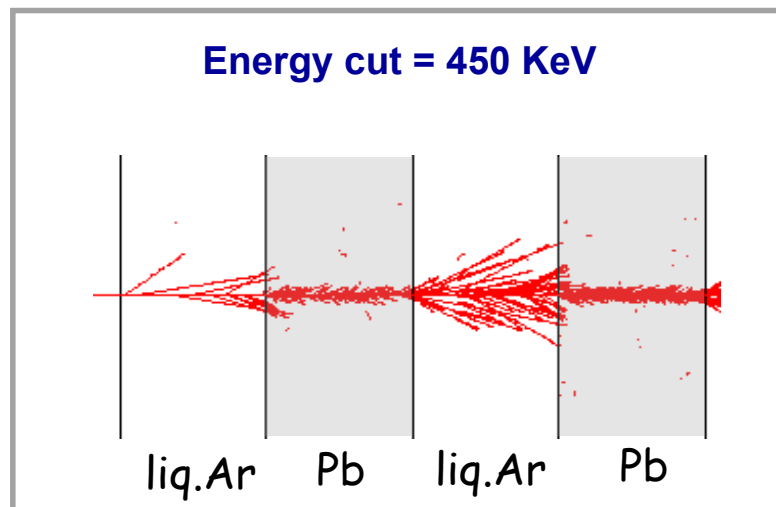
- GiGa modular physics lists used to build the PL to be ‘run’
  - allows dynamic loading (via configuration options) of particular physics “sublists” (a.k.a. builders)
  - increases flexibility and make changes easier for in depth investigations
  - Templated factories with instantiation of concrete builder constructor
- Physics List to run is specified in Gauss() configurable
  - ensure consistent set of factories are called
  - various combinations for EM and Hadronics are possible

```
Gauss().PhysicsList = { 'Em': 'Opt1', 'Hadron': 'FTFP_BERT',  
                        'General': True, 'LHCb': True, 'Other': False }
```

# Geant4 production thresholds



- Geant4 has production thresholds for EM processes
  - Specify range (which is converted to energy for each material) at which continuous loss begins, track primary down to zero range
  - Create secondaries only above specified range, or add to continuous loss of primary for secondaries of less energetic than travelling the required range in the current material



G4 Range cuts applied:

10 mm for gamma's

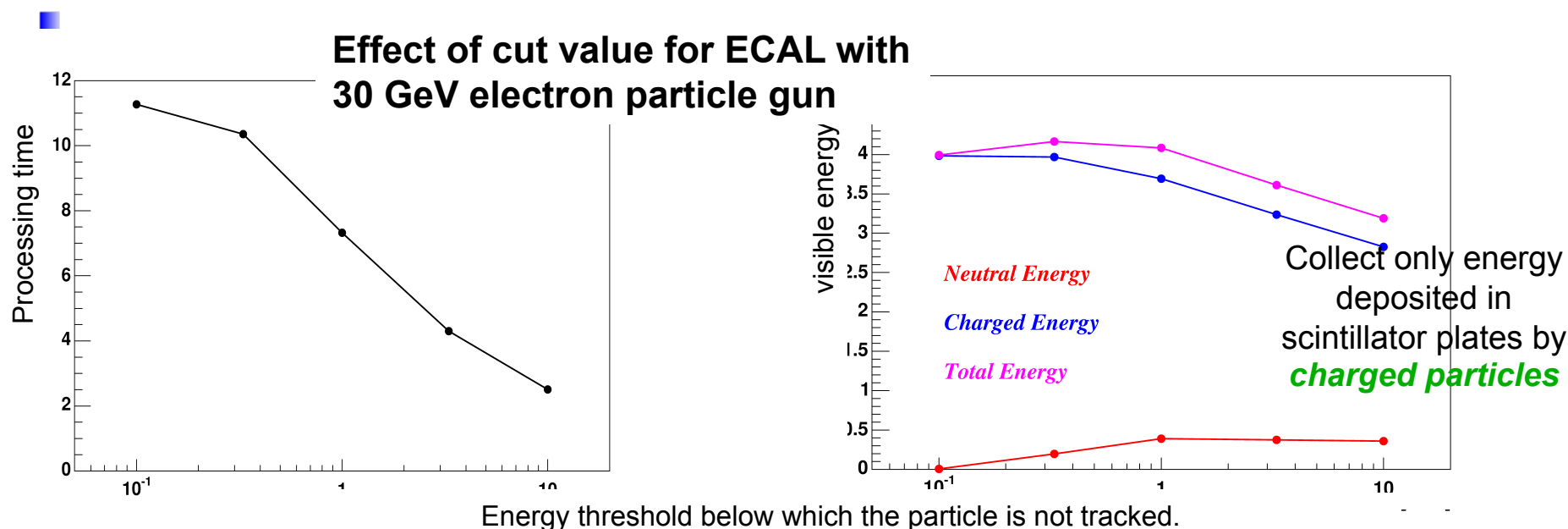
5 mm for e+

10 km for e- to have delta-rays off in trackers but affect dE/dx (see later): will change post-MC09



# LHCb tracking cuts

- Introduce tracking cuts on  $E_{kin}$  of particles of all type (special Gauss stepping actions)
  - Track particle until cut-off energy is reached, stop it at that point



- Possible also to set cuts per region
  - For example we turn on delta rays in RICH1 Aerogel (set in Simulation.xml)

# Geant4 Physics monitoring and validation

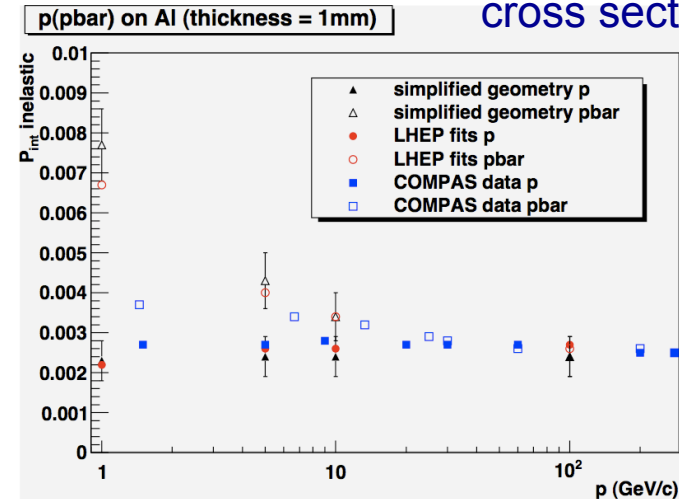


Every time a Geant4 version is changed:

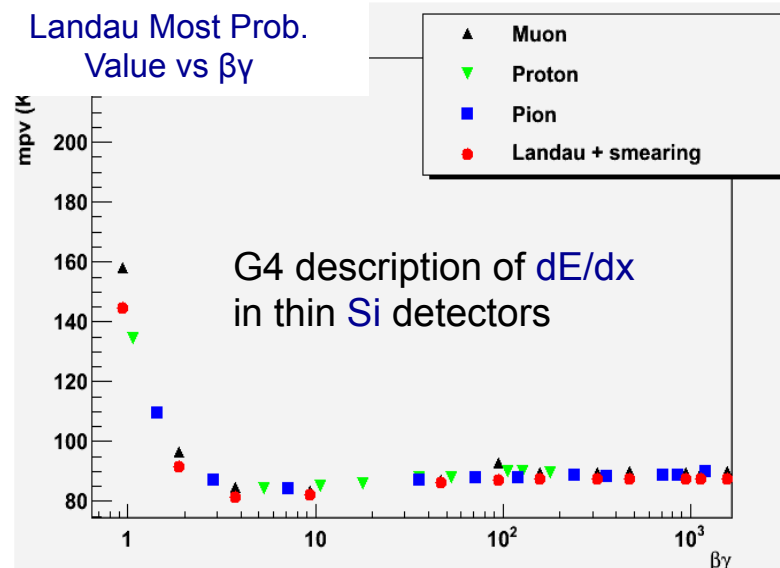
- main physics quantities are tested
- process-related simulation issues
- are kept under control

Validation done with Geant4.9.2.p03  
(PLs: LHEP, QGSP\_BERT, FTFP\_BERT).

Material interaction  
cross sections

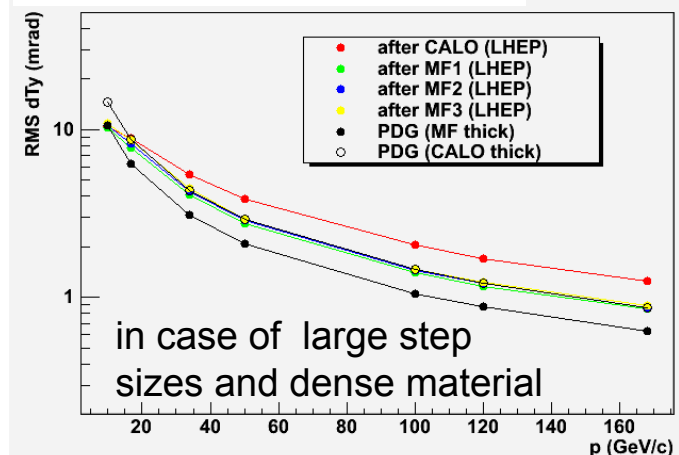


Landau Most Prob.  
Value vs  $\beta\gamma$



Angular deviation RMS vs p

G4 MCS



# Simulation of Detector response (aka digitization)



- Simulation of detector response transforming hits in sensitive detectors to produce digitized data and provide them in DAQ-like format is provided by Boole (separate application)

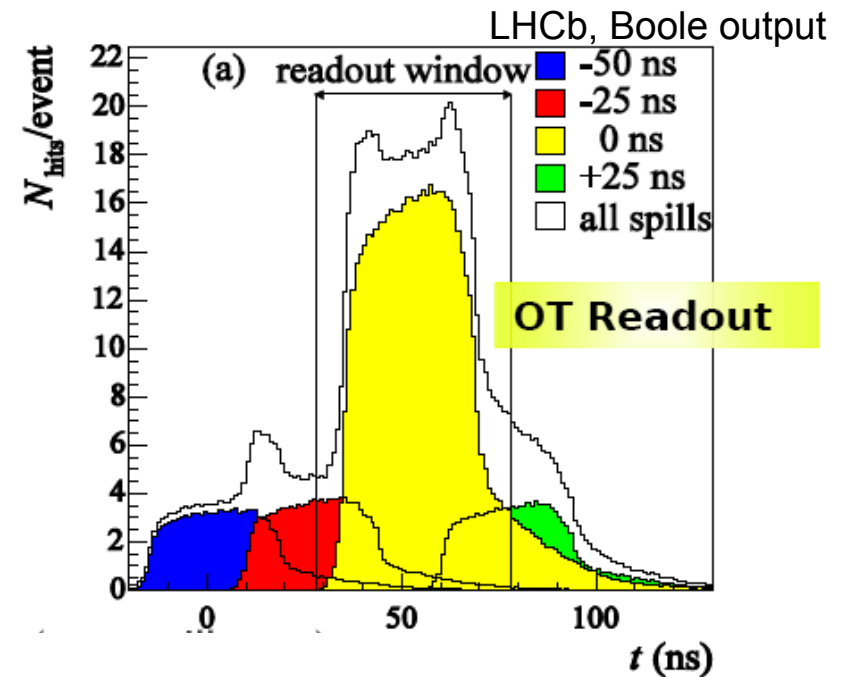
- Convenience
- Flexibility

- Each detector has its own detailed detector response simulation and imperfections

- Detections efficiencies and resolutions are adjusted according to test beam data
- Electronic noise and cross-talk for each specific detector are added
- Time information is correctly taken into account by all detectors

- Handling of spill-over effects (data from adjacent beam crossings)

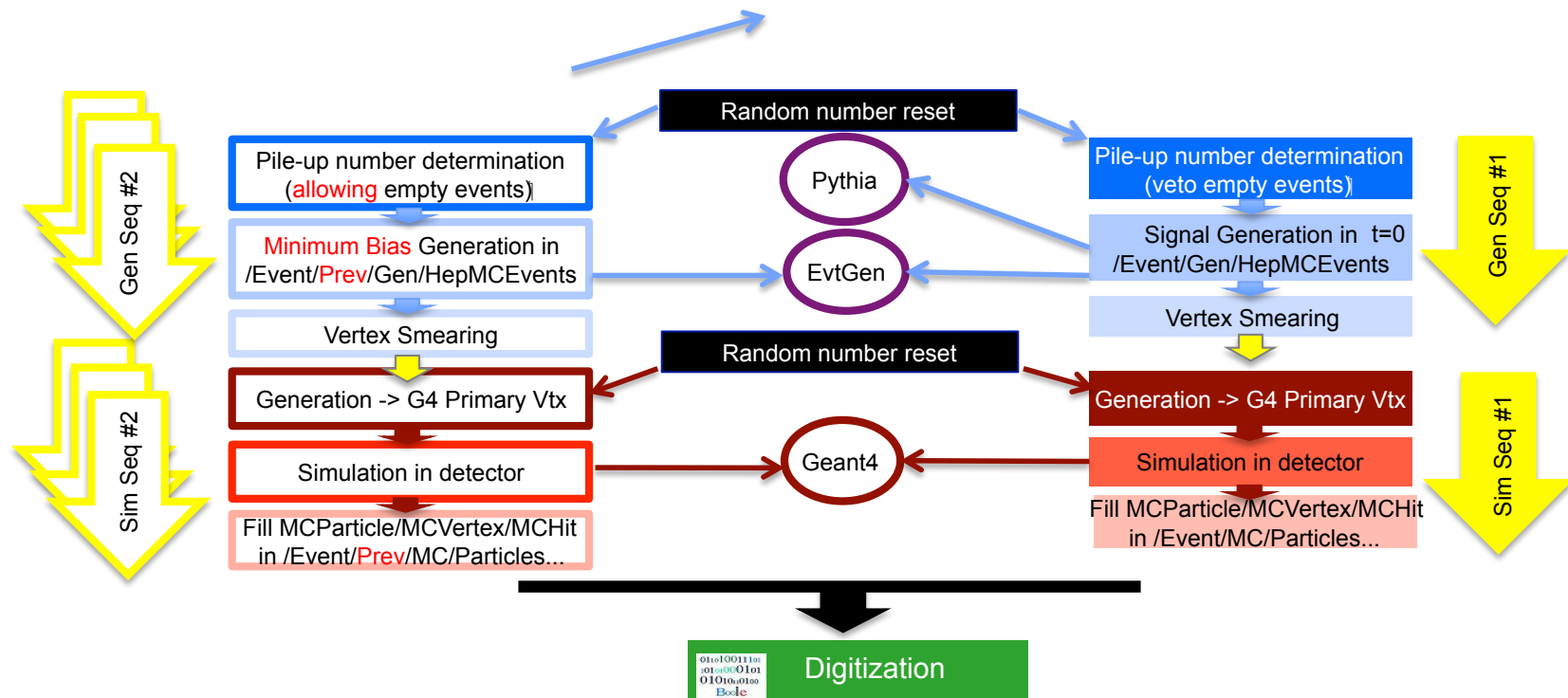
- A detector is sensitive to previous or subsequent bunch crossings depending on its electronics



# Spill Over Events in Gauss



- generate **spillover events in Gauss** in a single file and a single job
- there is a **single instance of Pythia, EvtGen, and Geant4** handling main event and spill-over events



# Reliability of the simulation



- Recurrent question is “What reasonable claims can be made for the upgrade, given the degree of realism (or not) that we see in the current MC?”
- MC tuning Campaign in progress
  - Real data is the reference to tune to
- Once the tuning is validated it becomes the reference for the simulation to keep or improve on
  - Must ensure the setting are fed back to the productions for the Upgrade
  - Simulation software and physics validation on a possible validation test infrastructure

# Understanding the simulation as a whole

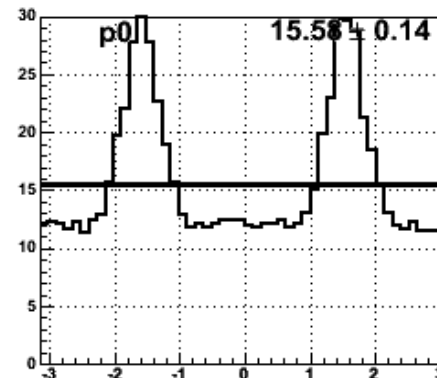
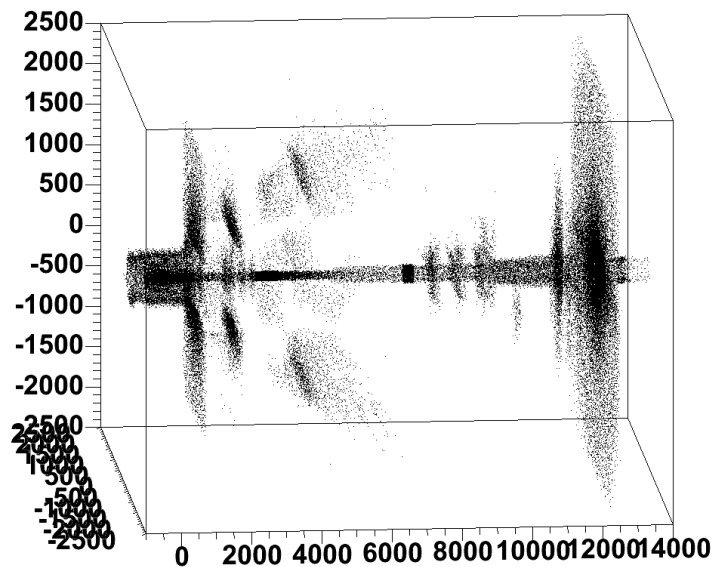


- To understand in depth the simulation, the information about the history of what happened in the detector is very important (trajectories and processes originating them)

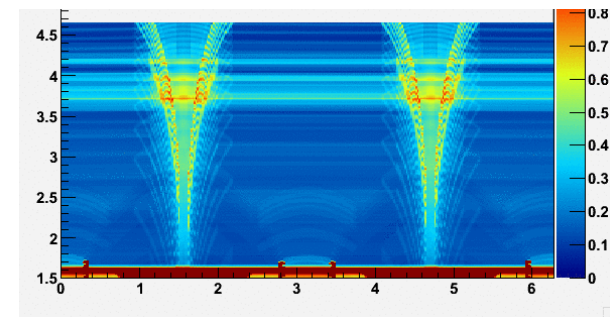
## Radiation length as seen by photon conversion

After Vertex Locator

Where particles originate in the detector



and match it with what seen in data and what put in simulation



- MC truth filled with both generator information and results of tracking through detector at the end of a Geant4 event

- LHCb::MCParticles

- What type, where created, energy and direction

- LHCb::MCVertices

- Initial pp interaction and all subsequent decays, with originating process identifier

- LHCb::MCHits (LHCb::MCCaloHits, LHCb::MCRichHits)

- Where each MC particle interacted with a sensitive volume
- Several different types for different detectors

*Relationship between them holds the tree structure*

*Actual trajectories of particles are forgotten*

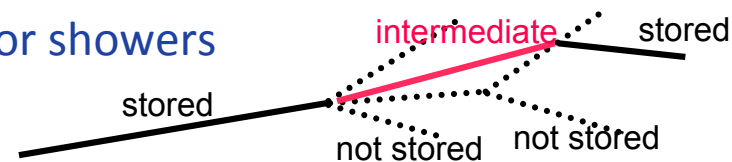
# The Monte Carlo Truth



- MC truth (MCParticles/MCVertices and their relationship) filled with both generator information and with results of tracking through detector at the end of the Geant4 processing of one event
  - Link from other classes ( MCHits for example, but also possibility to associate them to reconstructed Tracks, physics Particles)
  - These classes are written out by the simulation and accessed in a variety of ways in successive processing
- Generator to Geant4 and MC truth
  - Pass only particles from generator to Geant4 which will interact with detector, that is to say particles with non-zero travel length.
  - All other particles are saved directly in MCParticle container, and the decay chains are restored at the end of the processing by Geant4.
- MCHistory (i.e. what happened during the tracking of particles) is essential to understand efficiencies and physics effects



- Geant4 does not have a tree structure to keep history
  - the only way to interact with tracks in G4 when a process occurs is in StepAction, unfeasible
- Introduced use of HepMC internally to Geant4 to provide such a tree structure
  - easier to fill MCParticle/MCVertex relationship while keeping dependency of LHCb-specialized Geant4 classes to LHCb event model to a minimum
  - we access a G4track to decide to “keep it” either when it is created or when Geant4 stops tracking it
  - decide a-priori what to store through job options
    - e.g. discard optical photons, keep all products of decays in detector, ...
  - introduced intermediate particle for showers
  - split electron before and after Bremsstrahlung and to keep or not the split in algorithm transferring to MCParticles
  - distinguish production mechanisms by having a process identification in the vertices



# What is stored about processes?



- The MC decay tree stores
  - Initial pp interaction
    - If more than one pp interaction called a pile-up event
  - Each primary and decay vertex with the interaction type
  - Long lived particles
  - Intermediate resonant states
  - All children of hadron decays from the generator phase
  
- Not stored
  - Low energy particles (<100 MeV) before the calorimeter
  - Particles from showers in calorimeter
  - Particles that do not have hits [unless from a hadron decay of the generator phase]

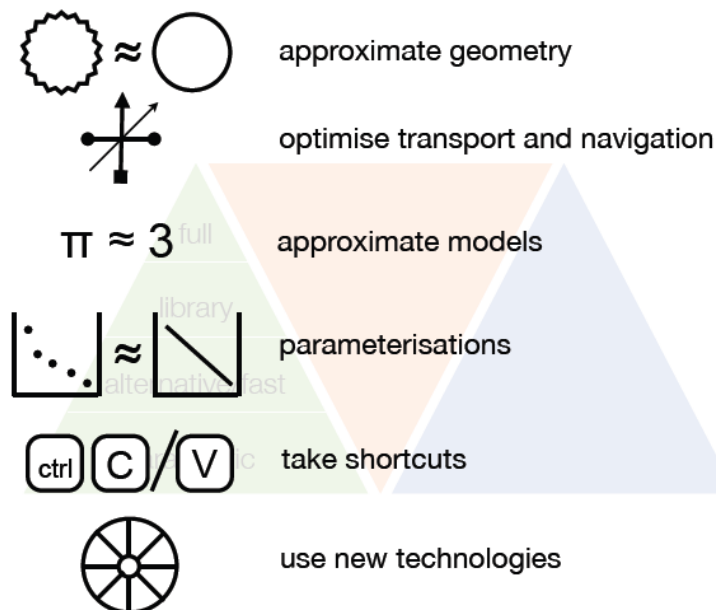
- A special algorithm (“OutputStream”) runs at the end of each event to select transient event data to be written to the persistent output file
  - Defines content of output file
  - Configured by “Configurables” in GaudiConf package
    - SimConf.py, DigiConf.py, DstConf.py
  - Called by main application Configurable
    - Gauss(), Boole(), Brunel(), DaVinci()
  - Several pre-defined contents are available:
    - gen: Gauss().Phases = [“Generator”]
    - xgen: Gauss().Phases = [“Generator”, “GenToMCTree”]
    - sim: Gauss default
    - digi Boole default
    - xdigi Boole().DigiType = “Extended”
    - dst Brunel default
    - xdst Brunel().OutputType = “XDST” (needs xdigi as input)
    - (sdst) Brunel().OutputType = “SDST” (only for real data production)
    - (mdst) Stripping output, several streams defined, only real data

- Gauss, hence Geant4, has been used for the production of simulation samples in the experiment since 2004
  - Billions of events produced in various MC Data Challenge to commission the experiment software and prepare the analysis
  - MC samples produced in the last year to reflect the operation conditions and provide input to physics analysis
- Production of millions of events in a distributed environment require a stable application but with fast technical patches
  - Decouple physics from software issues as fixes for crashes and memory leaks or CPU time improvements
- Performance of the simulation may limit the amount of MC samples that can be produced within the computing resources of the experiment

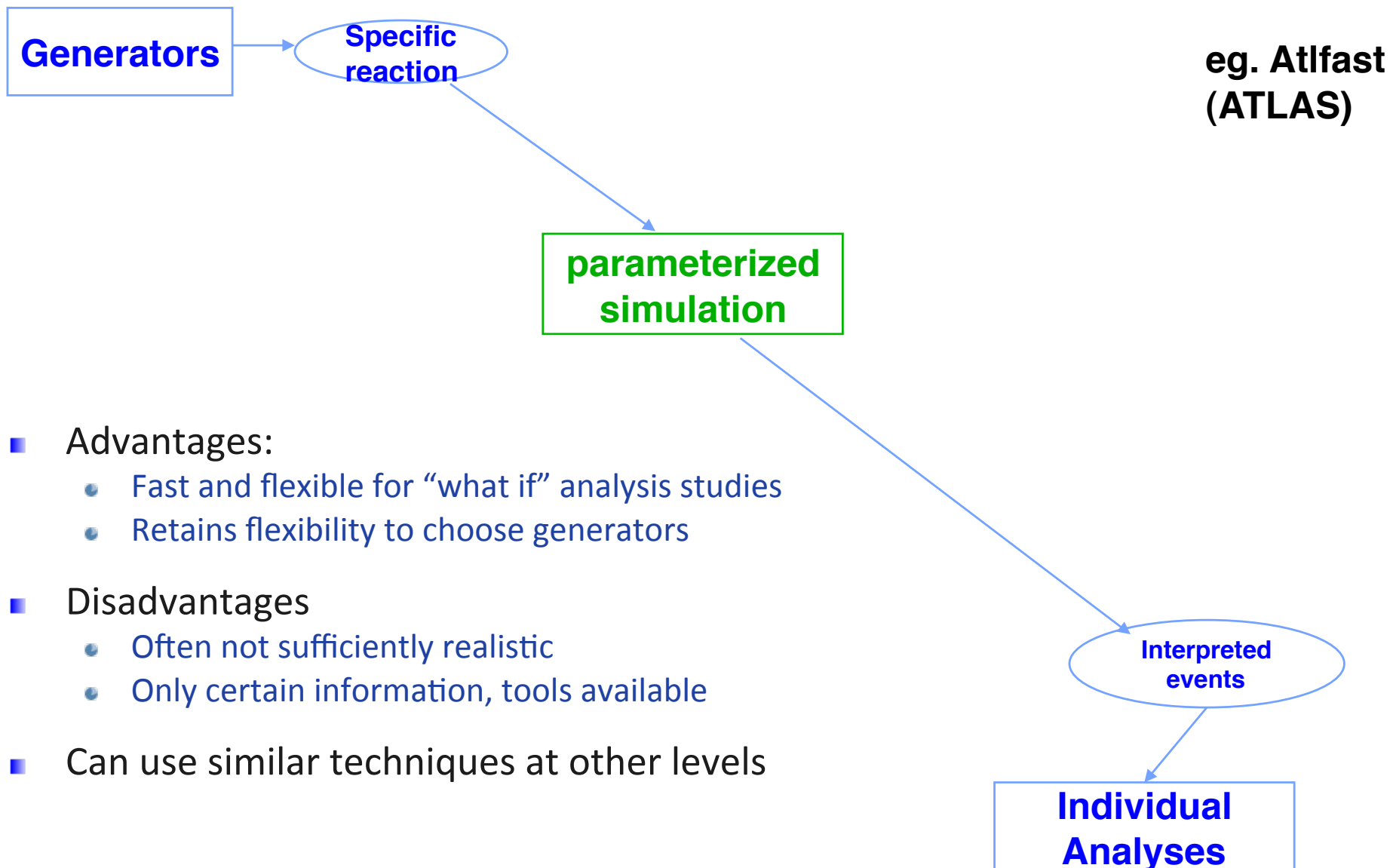
# Fast simulation

- ATLAS and CMS have heavily invested in Fast simulations in recent years
  - A way to cope with large MC statistics
- Of interest for physics analysis with the upgrade LHCb?
  - Bound to have tradeoffs:  
How to decide what is good enough?
- There are different levels of fast simulations
  - Could already do a smeared MC plugging together existing tools...

Fast simulation ❖ Ways to speed up simulation



# Simplify simulation by crossing levels



- Advantages:
  - Fast and flexible for “what if” analysis studies
  - Retains flexibility to choose generators
- Disadvantages
  - Often not sufficiently realistic
  - Only certain information, tools available
- Can use similar techniques at other levels

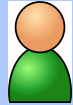


- Statistics required of the same order of real events
  - for LHC feasible for signal ( $O(10)$ - $O(10^6)$ /year), even many times more.
  - while impossible for all pp collisions ( $O(10^7)$ - $O(10^8)$ /sec) → months of running on the GRID to generate few seconds generic b events of LHC(b) data taking
- Simulations are the most CPU time consuming applications of HEP experiments
  - in LHCb the simulation takes **50 + 1 KSI2k sec/event (Gauss + Boole)**
  - while the reconstruction takes **2.5 KSI2k sec/event**
- Transport through the detectors is the longest
  - in LHCb: generator phase for most of the events types is  $1/100^{\text{th}}$  of simulation phase
- The CPU time depends on the complexity of the primary events
  - LHCb: minimum bias **20 KSI2k sec/event**
  - generic b events **50 KSI2k sec/event**



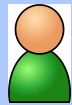
- In the LHCb experiment a wide variety of Monte Carlo samples need to be produced for the experiment physics program and require a production system and GRID resources to simulate events
- Procedures based on common infrastructures have been setup to handle Monte Carlo productions centrally
  - A numerical **Event Type id** has been devised to facilitate the configuration of the simulation application
  - Monte Carlo productions are a **customized type of productions** centrally handled by the Production Team
  - Deployments of new event types are managed through standard LHCb distribution software tools
  - The numerical event type id is also used to transparently **customize Production Requests** and to identify the samples produced
- Conventions allows transparent interplay of different elements

# Productions Steps and Players



Physics WG &  
Simulation software  
manager

**Step 1:** Preparation of decay description  
and configuration for new decay  
channel(s)



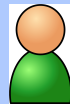
Release  
manager

**Step 2:** Release of new  
configurations and  
deployment on the Grid



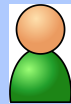
Simulation  
software  
manager

**Step 3:** Make the production  
system aware of the  
new event type(s)



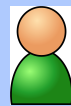
Physics WG

**Step 4:** Submission of MC  
request(s)



(MC) Production  
manager

**Step 5:** Production submission and  
follow up



Physicists

**Step 6:** Retrieval of data samples  
produced for analysis

# Event types and configuration



- In LHCb the majority of MC samples are proton-proton collisions with specific decay of b or c hadrons, but not only.
- EvtGen, an HEP-wide generator<sup>(\*)</sup> is used to model the decay of all particles
  - Default behavior is governed by a general DECAY.DEC table with all known decay modes for all particles
  - User decay files are used to force a specific decay for a signal particle via specific models
- LHCb has an extended version of the user decay files with a steering section to generate the specific configuration of the generators to be used at run time to produce a given sample
- The decays files and their automatically generated options reside in a dedicated data package, DecFiles, linked in at run time by the Gauss simulation application<sup>(\*\*)</sup>

(\*) <http://evtgen.warwick.ac.uk>

# Steering for automatic generation of options



**EventType:** Unique numeric 8 digits identifier based on the nature and topology of the decay

**Descriptor:** Details the decays in the file

**Cuts:** Generator level cuts. Each is implemented in a C++ class residing in a dedicated package

**NickName:** Short mnemonic unique and must match the file name

**Documentation:** Documentation about the decay file. It will appear with the provenance information in a documentation table on the web automatically made at release time

```
# EventType: 11114005
#
# Descriptor: {[[B0]nos -> mu+ pi+
-> K+ pi-)]cc, [[B0]os -> mu-
-> K- pi+)]cc}
#
# NickName: Bd
#
# Cuts:
#
# Acceptance
#
# John Doe
# John.Doe@mail.address
# 20110928
```

All used by a script run on demand or at  
release time to produce the options to be  
used by the simulation

Guidelines for all of them!

# Release of new Event Types



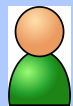
- Decay files for new event types are made by physicists and added to the SVN repository of the package
- DecFiles managers check that rules are respected. Automatic test in nightly builds to verify new event types can be processed.
- Release of DecFiles package asynchronous from that of the Gauss simulation application
  - Major released version number used to ensure compatibility
  - Version to be used specified in production system

# MC requests workflow



Physics WG  
Liaison

Collects needs and submit  
requests  
New/Submitted



Technical  
Expert

Verification of technical  
consistency

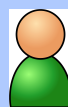
Tech Accepted/Rejected



Physics WG  
conveeners

Appropriateness of the request  
for the physics program and  
priority

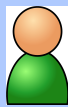
Phys Accepted/Rejected



(MC) Production  
manager

Submit Production after final  
verification and matching of resources

Active



(MC) Production  
manager

Statistics required Produced

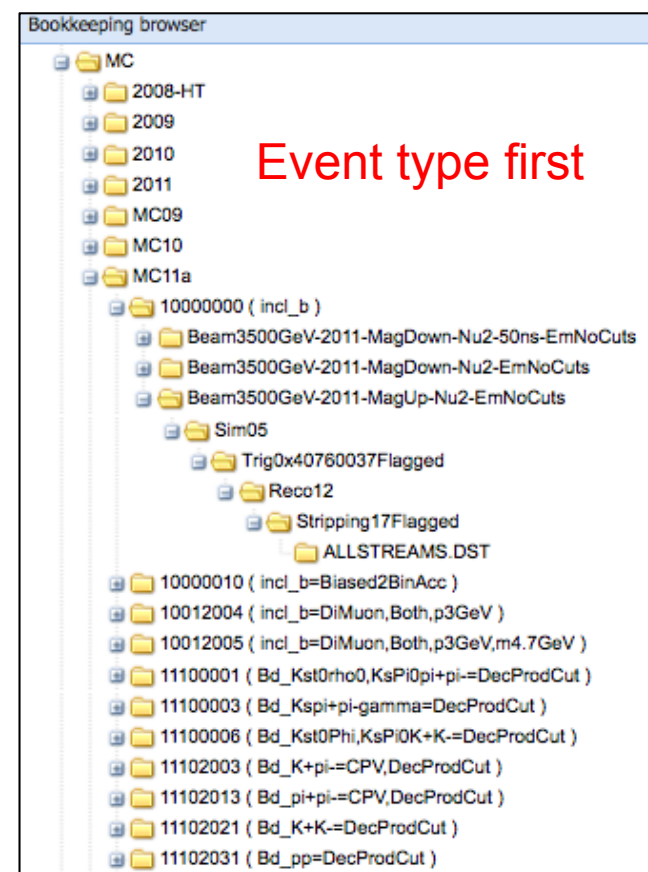
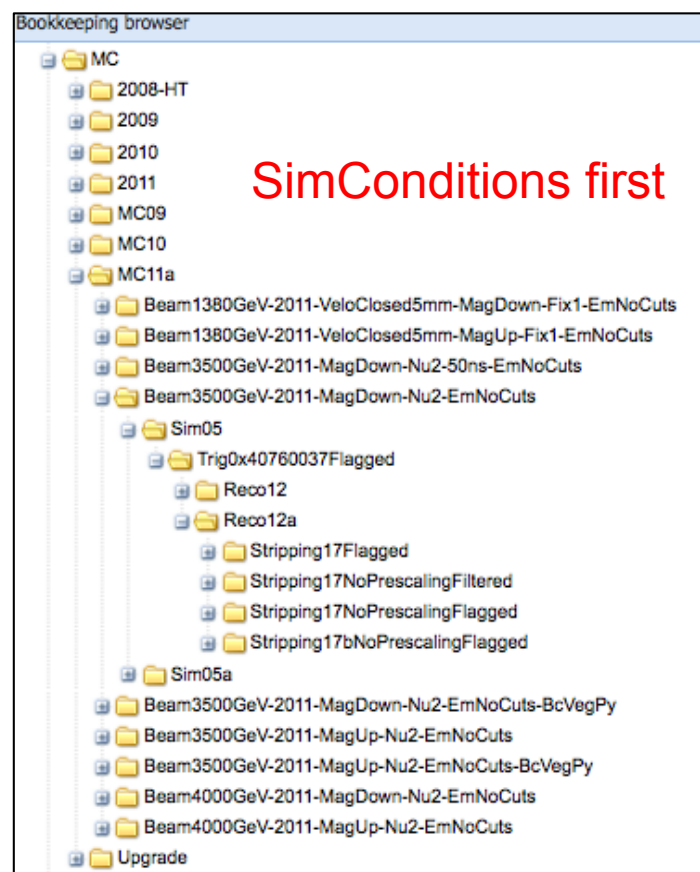
Done

# Finding MC datasets

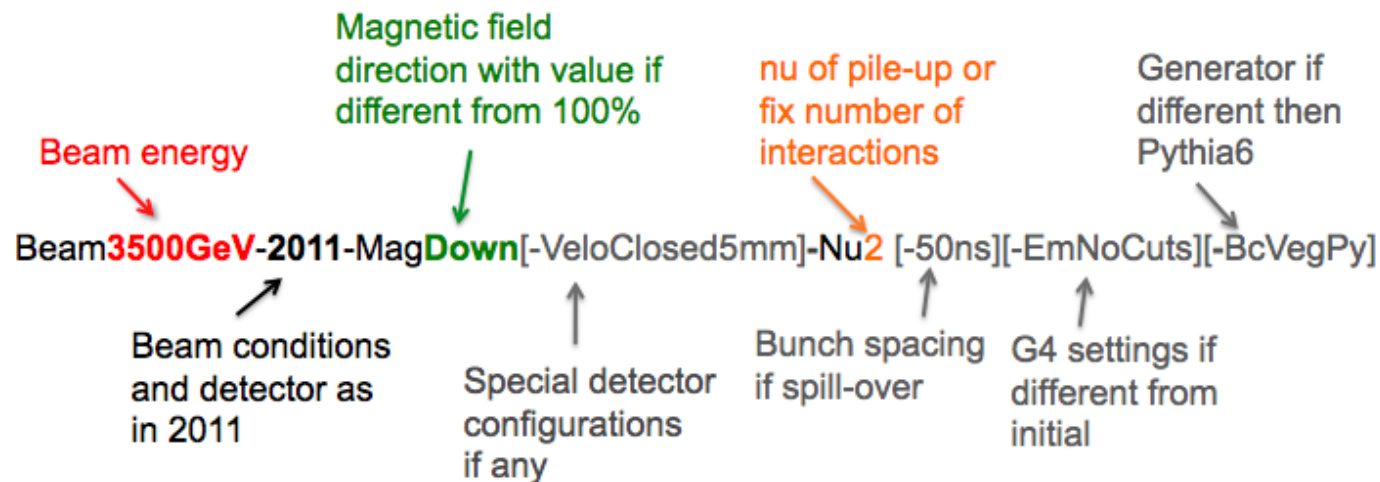


The metadata information of the tasks executed on the Grid will be uploaded to the Bookkeeping Metadata catalog at the end of the job as provenance of the data

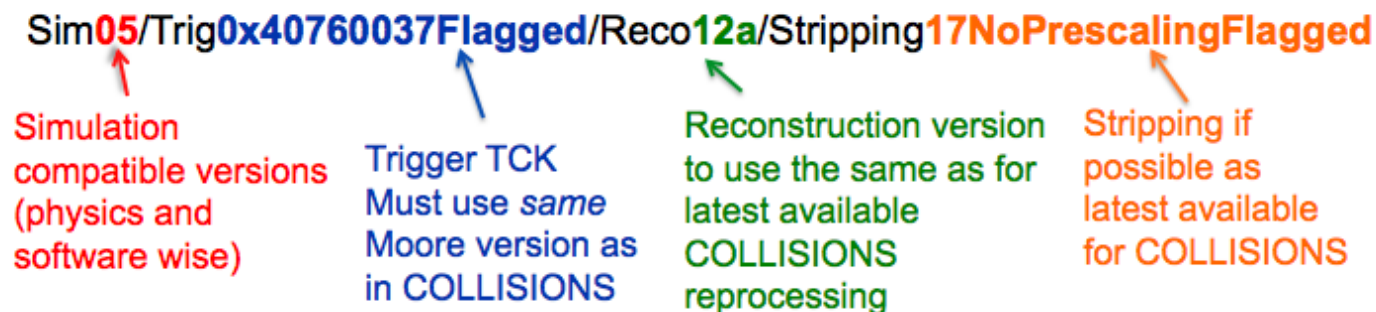
The whole of LHCb has access to all MC samples via the Bookkeeping



## ■ in the Simulation Conditions



## ■ and in the Processing Pass





- Another type of simulation is needed to evaluate environment detectors and electronics will experience
  - and to evaluate residual isotopes activities and ambient dose for maintenance and disposal of the detectors
- Performed entirely with FLUKA with a dedicated detector description.
  - FORTRAN-based with ASCII input for detector description
  - Couples low energy neutron transport and particle transport
- Many requests are coming in from upgrade detectors
  - Done by Matthias Karacson that also maintains the code
  - Some FT studies done by Neus March Lopes
  - Contribution by Vasily Kudryavtsev in particular tool to access the results

# Further information (1/3)




Web page (LHCb Home → Computing → Gauss)

<http://lhcb-release-area.web.cern.ch/LHCb-release-area/DOC/gauss/>

**THE GAUSS PROJECT**

HOME      RELEASES      LATEST RELEASE      PACKAGES

 **Welcome to the GAUSS project website**

*Gauss* is the simulation program of the LHCb experiment, based on the [Gaudi framework](#) and on the [LHCbSys](#) classes.

Gauss mimicks what will happen in LHCb to allow understanding of its experimental conditions and its performance. It integrates two independent phases:

1. A *Generator Phase* consisting of the generation of the p-p collisions and the decay of the particles produced
2. A *Simulation Phase* consisting in the tracking of the particles in the detector and simulating the physics processes occuring in the experimental setup

Gauss is interfaced to various specialized packages available in the Physics community for the *Generator phase* (e.g. [EvtGen](#) for b-decays) and makes use of the Geant4 toolkit for the *Simulation phase*. For more specific information on each of the phases and the libraries used follow the links provided above.

Gauss is structured as a [CMT](#) project containing an application package, also called **Gauss**, and a set of packages (component and linker both) specific to the physics simulation of the experiment and constituting the whole of the application. The application package provides the job options and main program for running the Gauss application. The default options are the configuration used in Production but for those specific to a given job (i.e. number of events, event type). Options for some different configuration of the application are provided. The configuration for different event types are provided in the [DecFiles configuration package](#).

**Announcements**

- **Latest current public release v38r5** (2010-05-09), with latest **release notes** and latest **doxygen** documentation.
- **Latest MC09 compatible public release v37r8** (2010-01-26)
- **Latest DC06 compatible public release v26r0** (2008-02-04)

To have access to all the available versions, please follow [this link](#).

**Production**



TWiki > LHCb Web > LHCbComputing > LHCbSimulation (28-May-2010, GloriaCorti)

## Simulation in LHCb

- ↓ [Introduction](#)
  - ↓ [Coordinators](#)
  - ↓ [Gauss Web](#)
- ↓ [Documentation](#)
  - ↓ [FAQ](#)
  - ↓ [Gauss Tutorial](#)
  - ↓ [Gauss User Guide](#)
  - ↓ [Information on Generators in Gauss](#)
  - ↓ [Information on Detector simulation \(Gauss and Boole\)](#)
- ↓ [Ongoing tasks/projects, open issues](#)
  - ↓ [Geometry and mis-alignment for simulation](#)
    - ↓ [Status for MC09 as used by Gauss v37r0, DDB tag head-20090330 a](#)
  - ↓ [Conditions for simulation](#)
  - ↓ [Samples for validation](#)
    - ↓ [Validation of generator phase](#)
    - ↓ [Validation of simulation phase](#)
  - ↓ [MC production histograms: monitoring tools](#)
  - ↓ [Radiation Length Scans using G4](#)
  - ↓ [Migration to HepMC2](#)
- ↓ [Current and future developments](#)
- ↓ [Bugs and features](#)
- ↓ [Next versions](#)

LHCb Twiki → LHCb Computing → LHCb Simulation

- <https://twiki.cern.ch/twiki/bin/view/LHCb/LHCbSimulation>

### Introduction

Gauss is the LHCb Simulation software, built on the [Gaudi Framework](#).  
It consist of a first phase where the events are generated (e.g. pp collisions at 14 TeV):





- Changes in between versions
  - Refer to release notes and example options in specific versions
  - Use latest released version (or development...)
  
- Mailing lists:
  - [lhcb-gauss@cern.ch](mailto:lhcb-gauss@cern.ch)
    - for general discussion and help from others using Gauss or detectors experts
  - [lhcb-gauss-managers@cern.ch](mailto:lhcb-gauss-managers@cern.ch)
    - to contact the Gauss core team



BACKUP

# Geometry validation



Every time the XML geometry description changes...

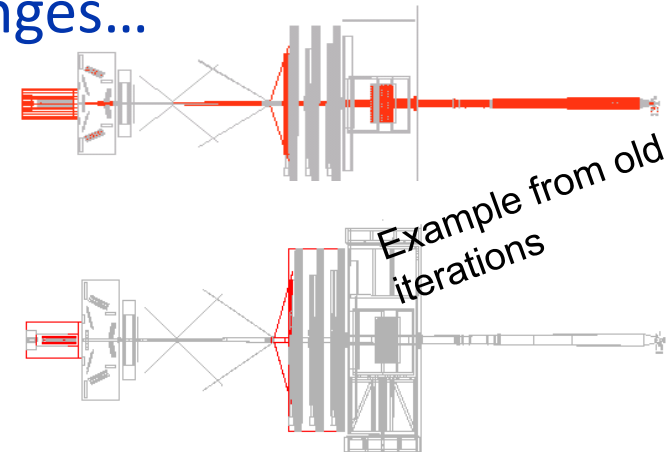
## Overlaps checks

Geant4 does NOT like volumes overlaps:

- Particle get stuck and event is (not) aborted
- Possible crashes

Must ensure when all mis-alignments are taken into account no overlaps are present

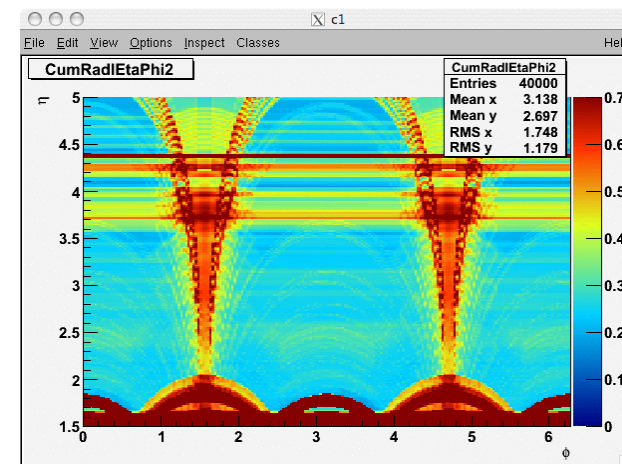
- Geant4 DAVID tool used to detect the overlaps between the volumes; converted to a graphical representation for visualization purposes (95% overlaps turned out to be due to precision problems)



## Material Budget evaluation

Radiation length evaluations performed periodically to compare the updates in the detector description:

- amount of material as seen by the particles at Geant4 Step-level
- comparison with material as modeled by the LHCb detector description (used in reconstruction)



# Geometry visualization

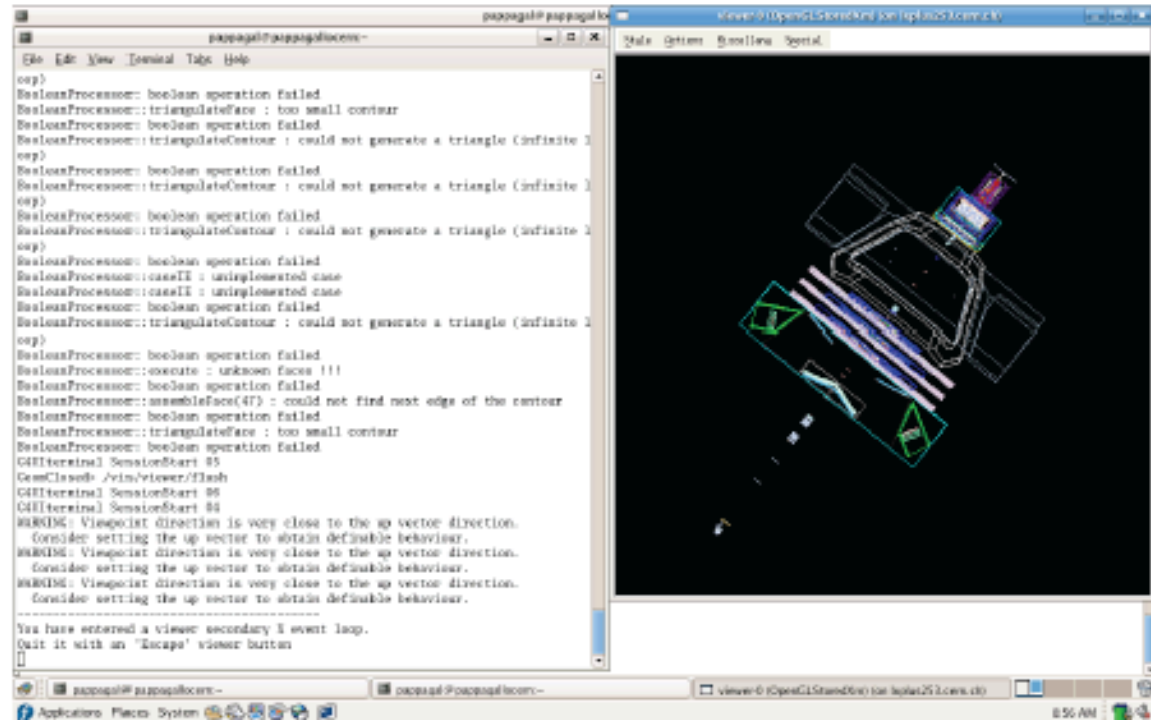


- Important to be able to visualize the geometry as seen by Geant4 when issues come up
  - Two Geant4 driver available from an ‘interactive’ version of Gauss

- **DAWN**: creates `.prim` files containing full geometry description. `.prim` files can be read by the same application (DAWN GUI interface, possibility to save high-resolution postscript files) or by other tools (DAVID)

- **OpenGL**: direct visualization of the geometry. Disadvantage: DAVID intersection debugger cannot be used with it

- **ASCIITree**: is a visualization driver that is not actually graphical, but that dumps the hierarchy as a simple text tree.



# List of available outputs



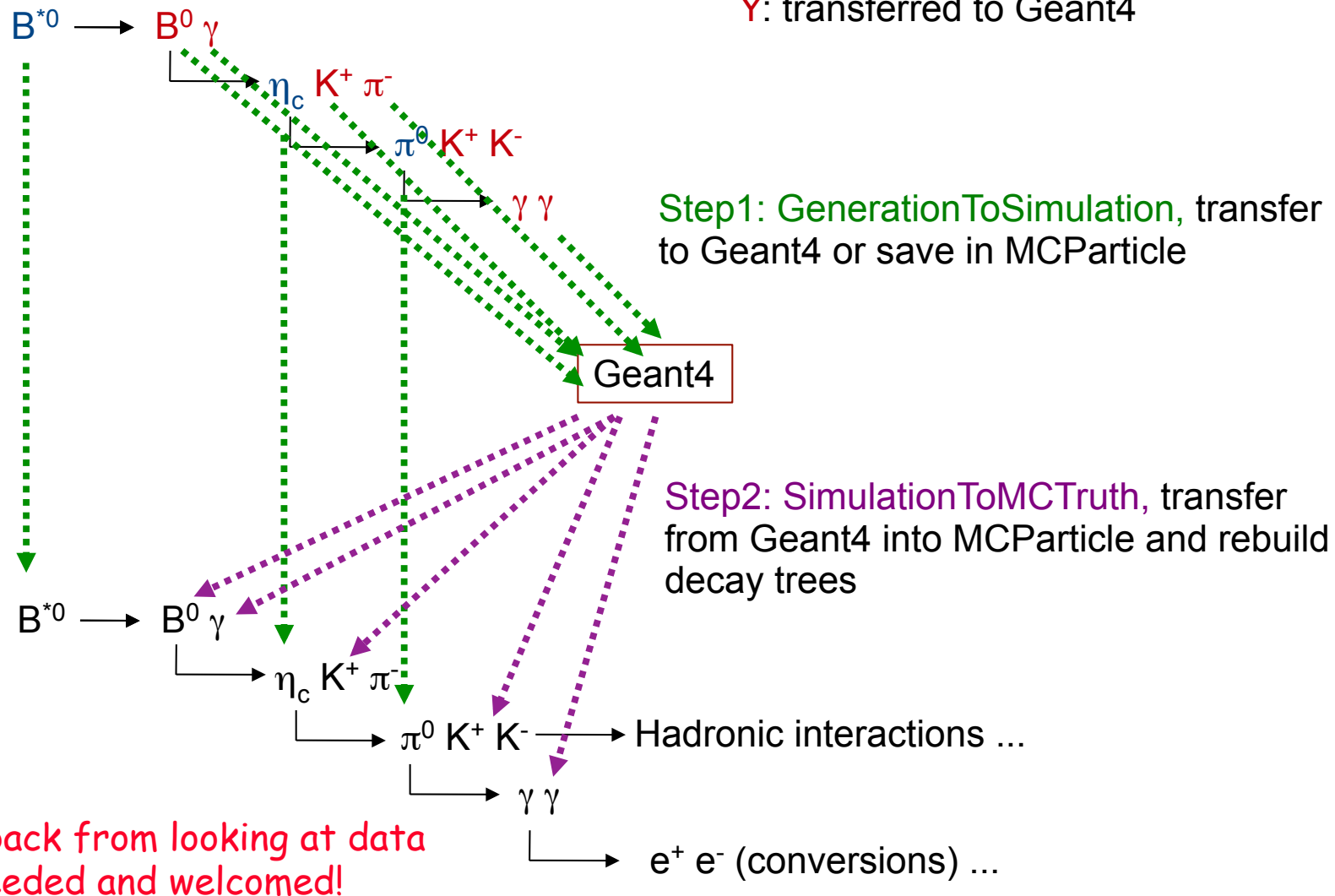
- **gen**
  - Generator files: Files with only the /Event/Gen tree, hence only generator information and the HepMC record. The complete hard interaction is available.
- **xgen**
  - Extended generator files: Files with the /Event/MC/MCParticles and /Event/MC/MCVertices tree in addition to /Event/Gen. To produce these files only the generator phase of Gauss has been run, i.e. the content in the MCtruth tree is a copy of the generator information after hadronization. In other words quarks and strings do not appear in the MCParticles.
- **sim**
  - Simulation files: Files resulting from the detector simulation. The configuration of the detector (all, velo open, calo only, dddb tag) is specified in the SimulationConditions. The MCtruth tree is filled with the information from the generator for primary particles and with the secondaries produced in the detector through transport. Hits for all detectors are also present (MCHits, MCRichHits, MCCaloHits) and additional simulation information as MCRichSegments. In case of simulations with spill-over the whole information is also available for the spill-over slots. The full /Event/Gen is copied.
- **digi**
  - Digitization files: Files containing the digitization of the simulation and the raw event as from the DAQ. They result from the Boole processing. After the Moore processing the trigger information is added. They include the MCtruth, hence the /Event/Gen and the /Event/MC/Particles and Vertices.
- **xdigi**
  - Extended digitization files: Files with the whole content of both sim and digi processing.
- **dst**
  - Dst files: Files resulting from the reconstruction of MC samples or real data. They include the RawEvent and the full reconstruction output. For MC they also include the MC truth, hence the /Event/Gen and the /Event/MC/Particles and Vertices. After stripping, also include particles forming signals selected by the lines that have fired.
- **xdst**
  - Extended dst files: Files with the whole content of sim, digi and dst (simulated data only).
- **sdst**
  - dst files for input to real data stripping: reconstruction output only, no MC truth, no RawEvent.
- **mdst**
  - MicroDST files: the smallest format containing reconstruction information of only particles forming signal selected by the stripping lines that have fired. MC truth not implemented.



# Example of how particles are given to G4

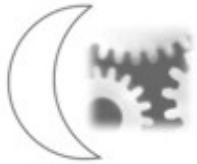


X: saved to MCTruth directly  
Y: transferred to Geant4



Feed-back from looking at data  
needed and welcomed!

# Gauss automatic software testing



- New releases are built with about  $\sim$ 2/month frequency, plus patches for production when needed
- **upcoming releases** and development versions are tested in the LHCb nightly system
- set of **Run Time tests** ( $\sim$ 30 QMTests) for **specific simulation tests cases**
  - fast debug of detector or physics problems
  - generator-only (signal events, different generators samples) and full chain tests (minbias)
  - run both development and production configurations

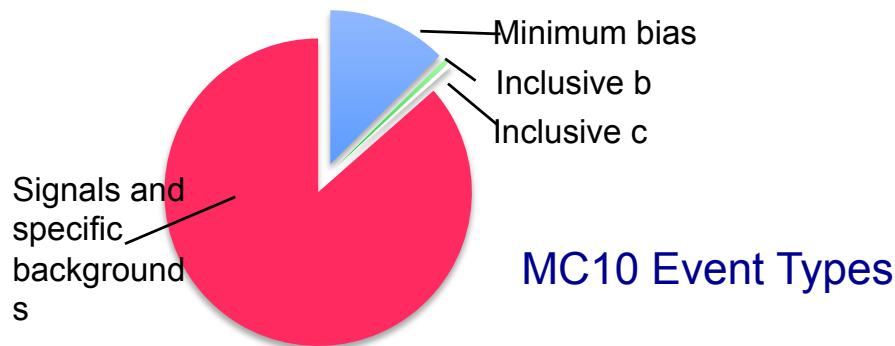
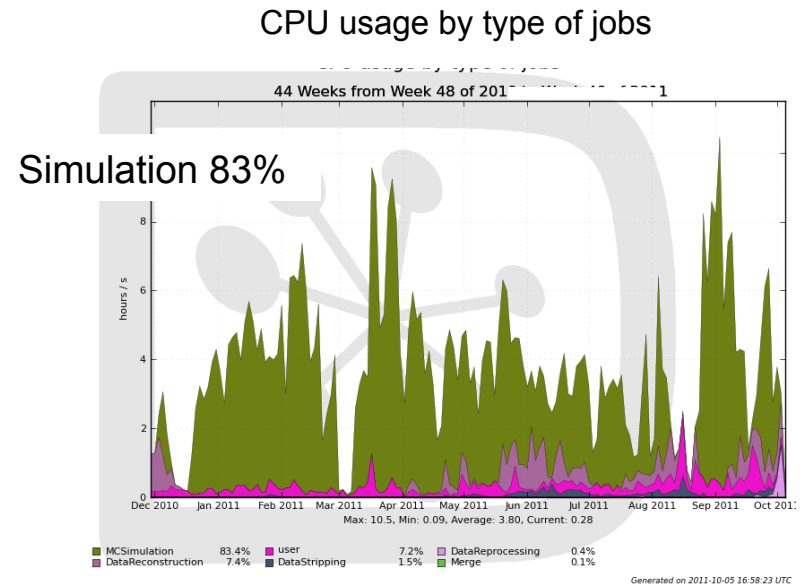
Wednesday Slot : lhcb-head - head of everything against GAUDI\_v22r4 and LCGCMT\_61

Project	Version	x86_64-slc5-gcc43-opt (Wed Oct 5 07:14 2011)	x86_64-slc5-gcc43-dbg (Wed Oct 5 06:05 2011)	i686-slc5-gcc43-opt (Wed Oct 5 05:54 2011)	i686-slc5-gcc43-dbg (Wed Oct 5 07:37 2011)	x86_64-slc5-icc11-opt (Wed Oct 5 06:55 2011)
Geant4	GEANT4_HEAD	build (2) tests	build tests	build (2) tests	build tests	build (65) tests
Gauss	GAUSS_HEAD	build (39) tests (13)	build (39) tests (13)	build (39) tests (8)	build (39) tests (8)	build (445) tests (13)

# Gauss in Production

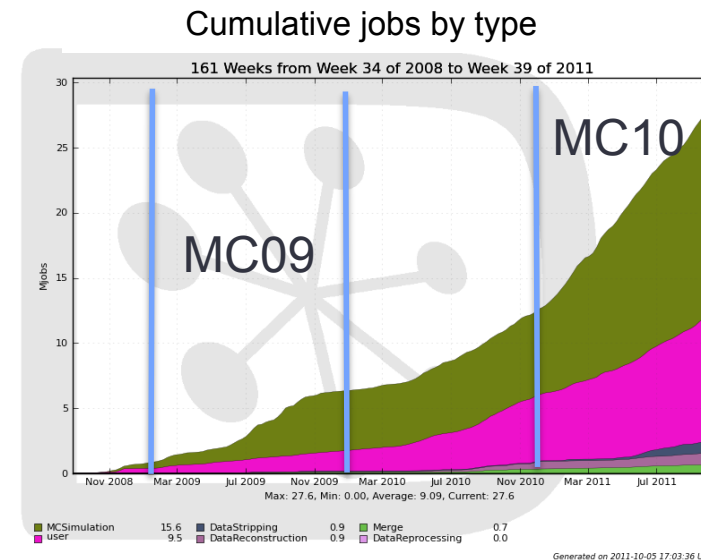


	Geant4	Total Events	Event Types	Total file size
DC06	7.1	598M	155	129TB
MC09	9.1.p03	2791M	165	198TB
2010-DEV	9.2.(p0X)	663M	165	99TB
MC10	9.2.p04	1169M	380	420TB
MC11	9.4.p02	~ 1 to 2 B	to start soon	



Production on slc5 32-bit  
issue of jobs 'hanging' but not due to particles stuck  
on 64 bit, for now on a back-burner

Stable memory consumption



# Request models and request



Event Types

Bookkeeping Metadata Catalog Browser

The screenshot shows the 'Bookkeeping Metadata Catalog Browser' interface. On the left, a tree view displays the directory structure under 'MC11a', including folders for 'Sim05', 'Reco12a', and 'Stripping17'. A green arrow points from the 'Event Types' starburst to the 'Bookkeeping browser' folder. The main window displays details for 'Request: MC11a Model - MD - Spillover - Trig and Strip flagged'. The 'Request' section includes fields for Name, Type (Simulation), State (New), Priority (2b), Author (gcorti), and MC Config (MC11a). The 'Simulation Conditions' section provides a description and parameters such as Beam energy (3500 GeV), Generator (Pythia), and G4 settings. The 'Processing Pass' section details three simulation steps. The 'Event' section on the right shows a list of event numbers and comments, including '10000000 - incl\_b' and '10000010 - incl\_b=Biased2BinAcc'.

- The Request Manager page allows MC liaison to create their simulation requests which will perform on the Grid 'cloning' existing models and only specifying the event types and statistics

- Eight digit number of type “**GSDCTNXU**” to uniquely identify each decay file, associated options and samples produced
- Convention and extensions established and documented in LHCb notes:
  - First six numbers describe the decay and the last two distinguish between similar decays

**G:** General event type and production scheme.  
**S:** Value based on the presence of certain particles.  
**D:** Number depends on the general features of the decay.  
**C:** Based on the number of charm hadrons and leptons.  
**T:** Number of stable charged particles:  $\rho$ ,  $\pi$ ,  $K$ ,  $e$  and  $\mu$ .  
**N:** Number of neutrals :  $K_S$ ,  $\lambda$ ,  $K_L$ ,  $\gamma$ ,  $n$ ,  $\pi^0$  and  $\eta$ .  
**X:** Used to distinguish between different decays that share the same first 6 digits (different **Decay part of the Nickname**)  
**U:** Used to distinguish between the same decay, but different model, cuts, options (different **Other part of the Nickname**)