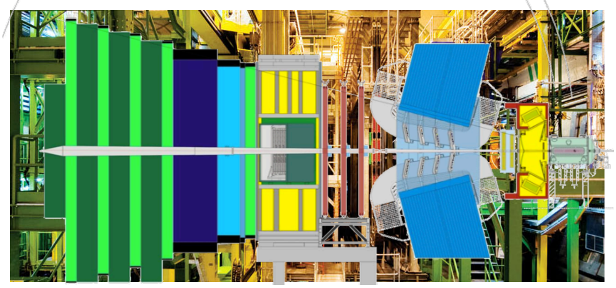


LPCC Workshop
"LHC Detectors Simulation: status, needs and prospects"
October 6^{th} – 7^{th} , 2011
CERN

Geant4 in the LHCb simulation application, Gauss

Overview of the application framework
Encapsulation of Geant4 and key aspects
Production issues

G. Corti (CERN)

Geant4 and LHCb software

- Geant4 is used in the context of the LHCb simulation application, Gauss
  - Some software choices on how to use it depend on this, other on functional requirements

- Gauss provides
  - Connection to event generators
    - Ensure consistency of decays
  - Event model for MC truth and Persistency
    - Access to snap-shots of processes to understand
  - Histograms, messaging

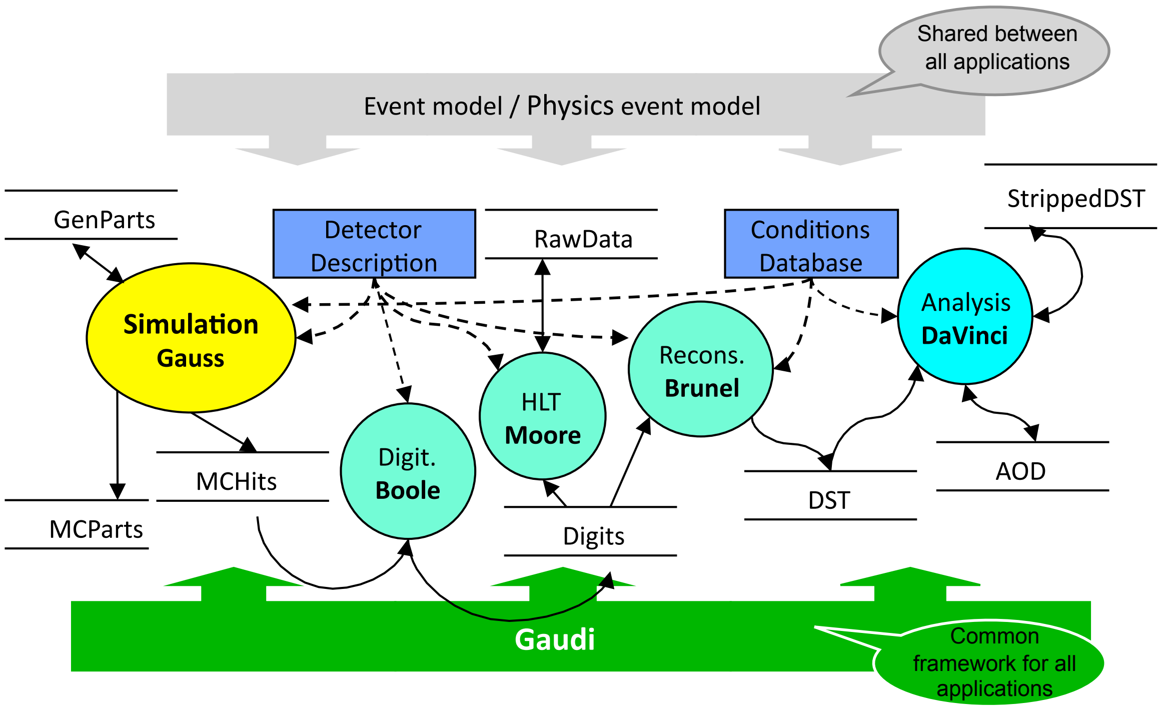
- Physicists in the experiment are shielded from Geant4 to different degrees
  - Different for different roles

The Gauss Application

Gauss, the **LHCb simulation application**, mimics what happens in the spectrometer to understand experimental conditions and performance. It provides:

- generation of proton-proton collisions
- decays of particles with special attention to B decays
- tracking of particles through 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 (digitization, reconstruction,…)

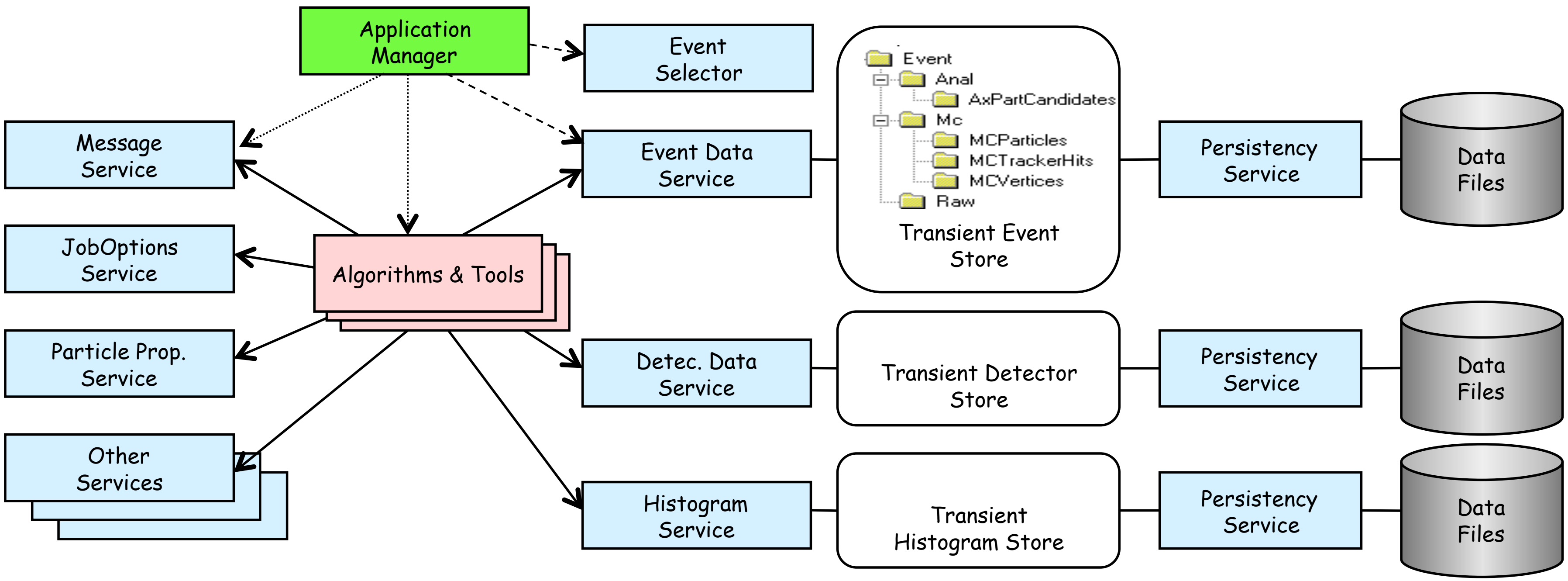


Gauss as a Gaudi Application

Gauss is based on the **Gaudi framework** and follow its architectural design

- Separation between “data” and “algorithms”
- Separation between “transient” and “persistent” representations of data
- “User Simulation Code” (**Gauss**) encapsulated in few specific places (*Algorithms*, *Tools*), callable and controllable from within the framework
- Well defined component “interfaces”

http://cern.ch/proj-gaudi/welcome.html

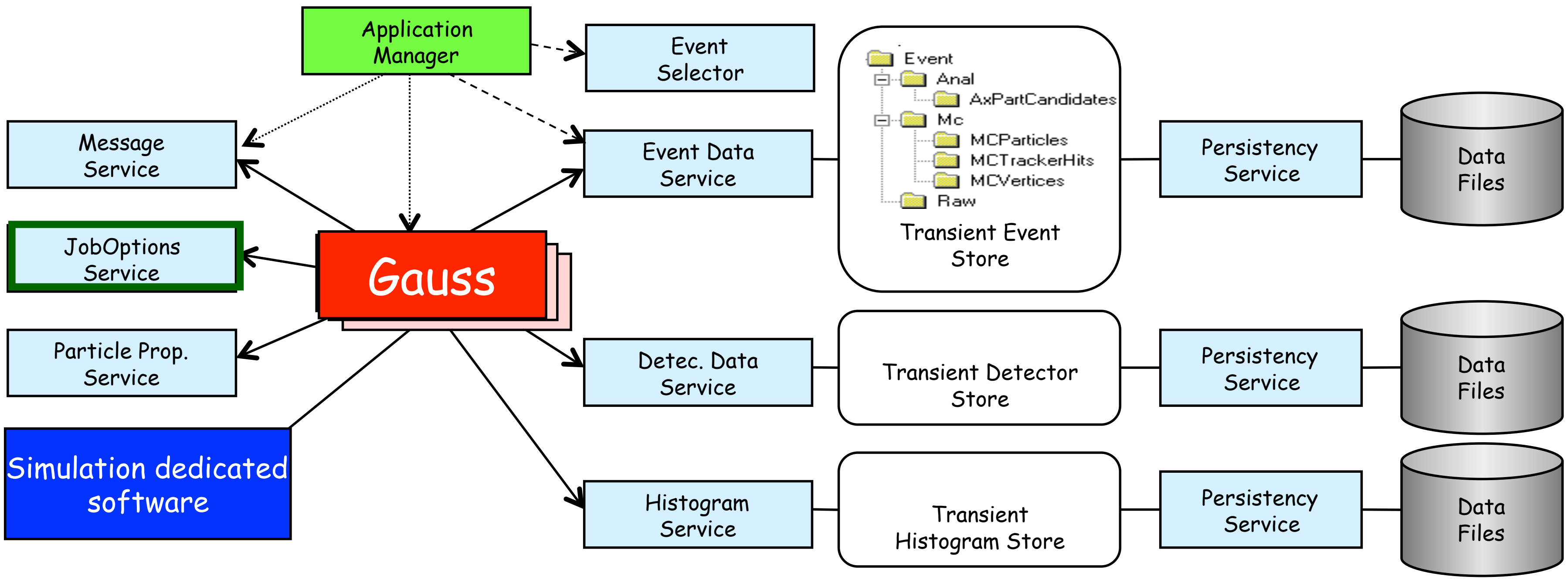


Gauss as a Gaudi Application

Gauss is based on the **Gaudi framework** and follow its architectural design

- Separation between “data” and “algorithms”
- Separation between “transient” and “persistent” representations of data
- “User Simulation Code” (**Gauss**) encapsulated in few specific places (*Algorithms*, *Tools*), callable and controllable from within the framework
- Well defined component “interfaces”

http://cern.ch/proj-gaudi/welcome.html

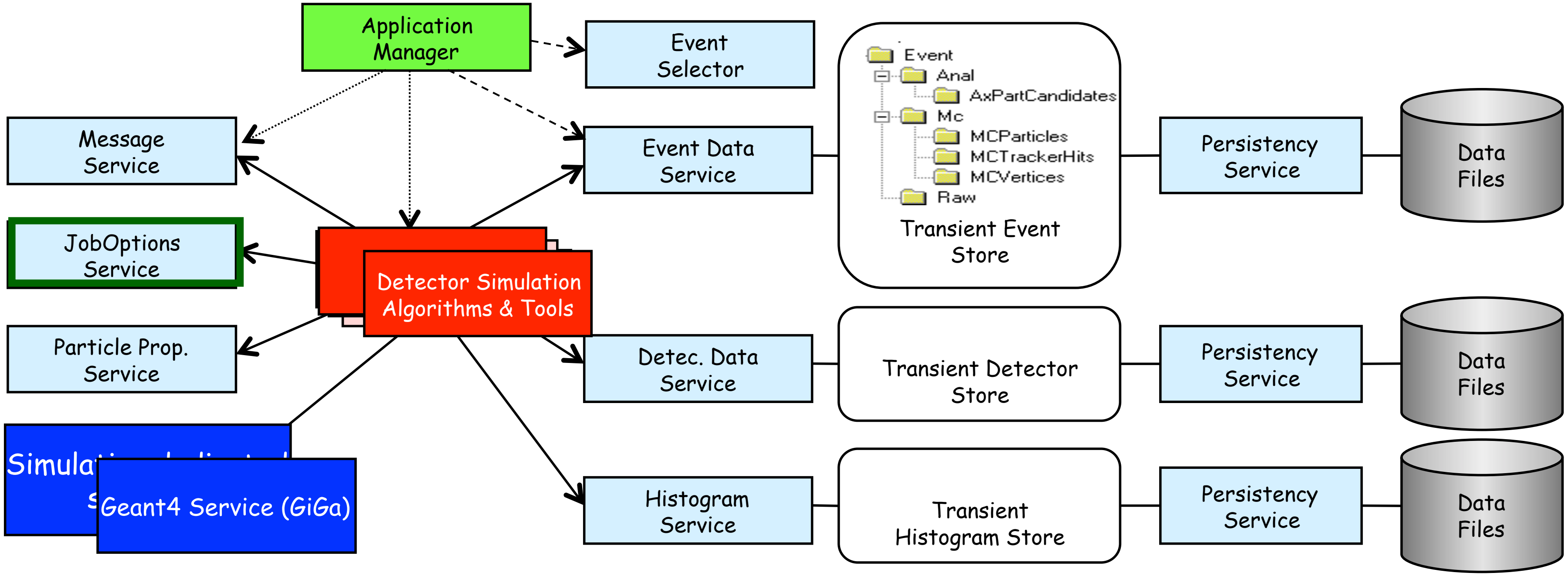


Gauss as a Gaudi Application

Gauss is based on the **Gaudi framework** and follow its architectural design

- Separation between "data" and "algorithms"
- Separation between "transient" and "persistent" representations of data
- "User Simulation Code" (**Gauss**) encapsulated in few specific places (*Algorithms*, *Tools*), callable and controllable from within the framework
- Well defined component "interfaces"

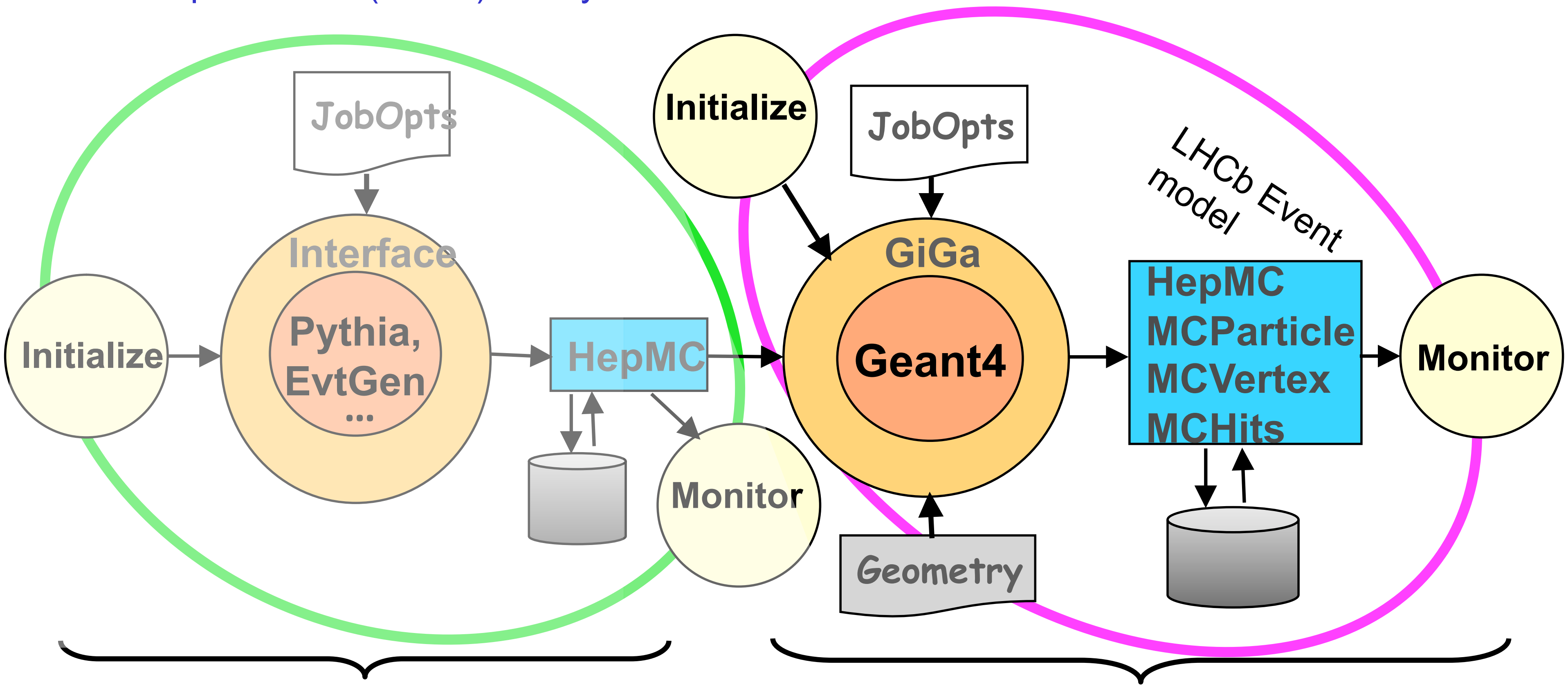
http://cern.ch/proj-gaudi/welcome.html



Gauss Structure

Two INDEPENDENT phases normally run in sequence in a single job

but each phase can (and is) run by itself



Event Generation
primary event generator
specialized decay package
beam conditions

Detector Simulation
geometry of the detector (LHCb → Geant4)
tracking through materials (Geant4)
hit creation and MC truth information (Geant4 → LHCb)

Application Sequence

Random number reset

Pile-up number determination (veto empty events)

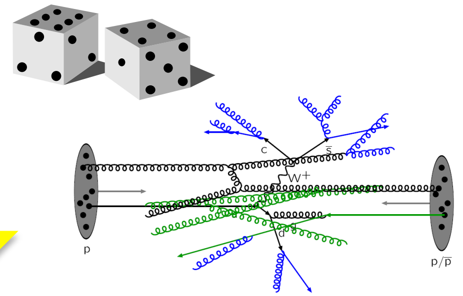
Pythia

EvtGen

Signal Generation in /Event/Gen/HepMCEvents t=0

Vertex Smearing

Generator Sequence



Random number reset

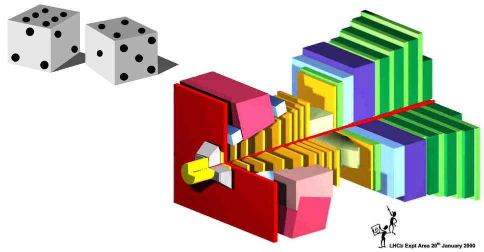
Generation -> G4 Primary Vertex

Geant4

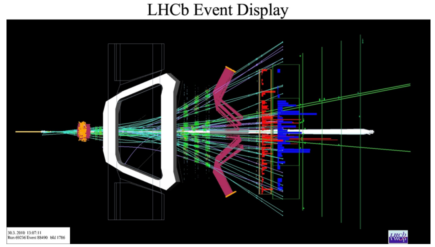
Simulation inside the detector

Fill MCParticle/MCVertex/MCHits in /Event/MC/Particles...

Simulation Sequence

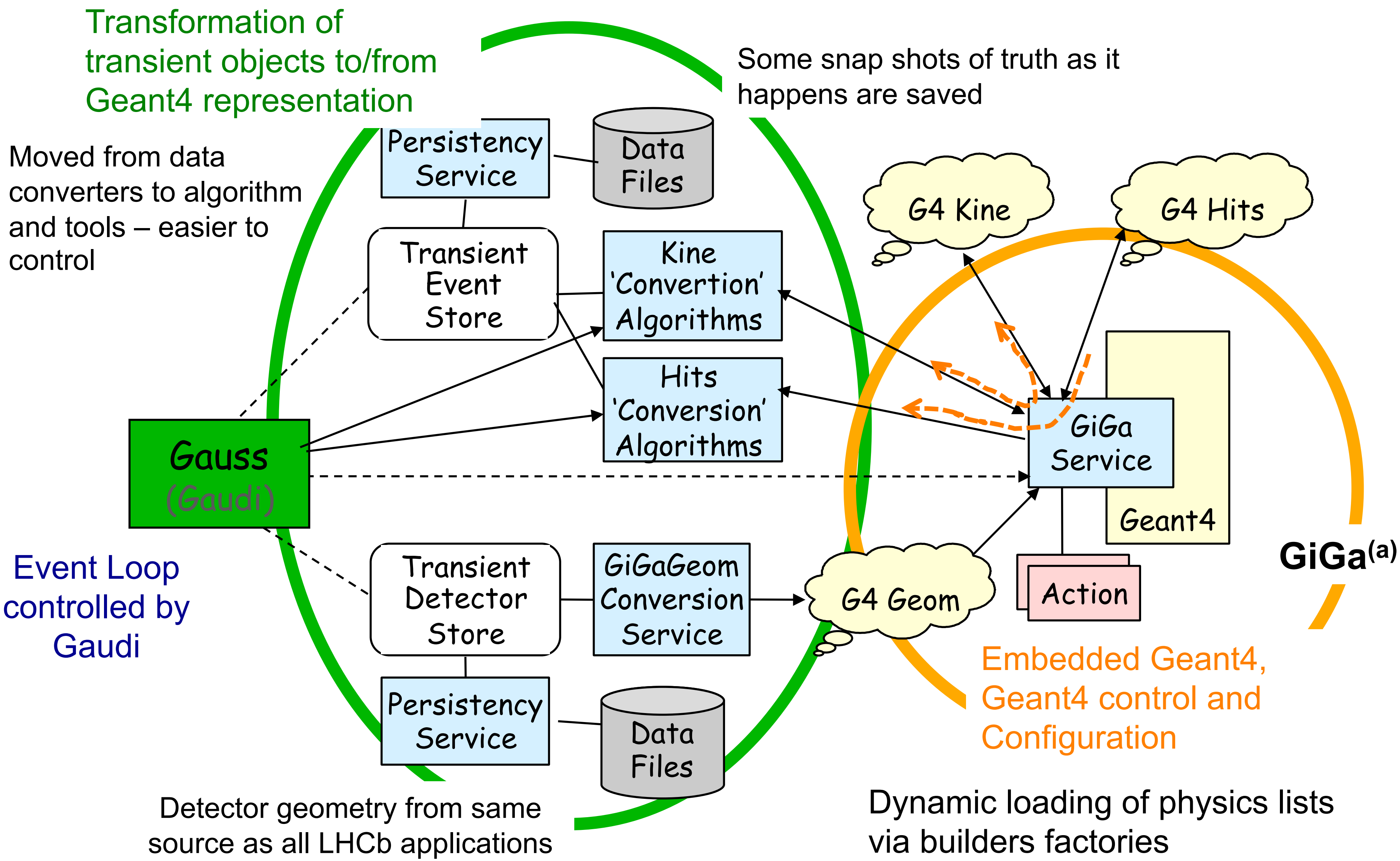


LHCb Event Display



Digitization

Encapsulation of Geant4 in Gauss



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

Gauss Python Configuration

The Gauss application is steered via a Python Configuration

- Basic option validation (types, names, etc.)
- Programming language (loops, logic, modularization, etc.)
- High level configuration (simple command for complex configuration tasks)

Python files

- OOP language
- the Python configuration is based on Configurables
- Configurables are special python classes built from the C++ components (Services, Algorthms, Tools…)
- each Configurable instance has got a name that is unique by construction

The Gauss configurable takes care of the sub-sequencing of the simulation phase

1. Passing the generator information to Geant4
2. Triggering the processing of the event by Geant4
3. Retrieving the MC truth and sensitive detectors responses from Geant4

and controlling the Geant4 settings for the event processing

Gauss Python Configuration

High level configuration with Gauss Configurable(s):

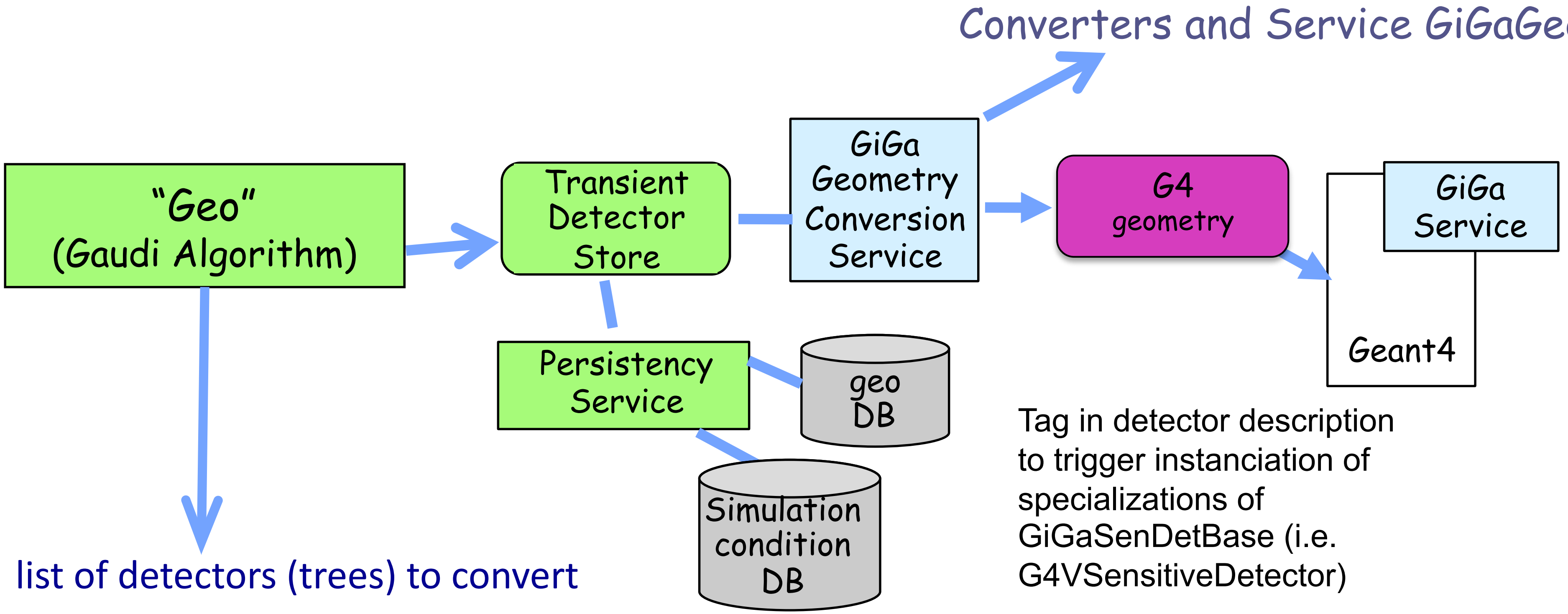
Automatic dispatching of the options to different algorithms and tools

Consistency checks can be performed,
e.g.
monitoring algorithms run only for simulated sub-detectors

```
# /***** User Gauss/Gauss ****************************************
# |-DatasetName             = 'Gauss'
# |-DetectorGeo             = {'VELO': ['Velo', 'PuVeto'], 'MUON': ['Muon'], '
# |                           (default: {'VELO': ['Velo', 'PuVeto'], 'MUON': ['Mu
# |-Histograms              = 'DEFAULT'
# |-BeamEmittance           = 1.0100000000000001e-06  (default: 7.0400000000000
# |-Phases                  = ['Generator']  (default: ['Generator', 'Simulatic
# |-Production              = 'PHYS'
# |-TotalCrossSection       = 9.1099999999999988e-24  (default: 9.7199999999999
# |-BeamCrossingAngle       = -0.00027  (default: 0.00032900000000000003)
# |-InteractionPosition     = [0.0, 0.0, 0.0]  (default: [0.0, 0.0, 0.0])
# |-PhysicsList             = {'Em': 'Opt1', 'GeneralPhys': True, 'Hadron': 'LH
# |                           (default: {'Em': 'Opt1', 'GeneralPhys': True, 'Hadr
# |-DetectorMoni            = {'VELO': ['Velo', 'PuVeto'], 'MUON': ['Muon'], '
# |                           (default: {'VELO': ['Velo', 'PuVeto'], 'MUON': ['Mu
# |-BeamSize                = [0.044999999999999998, 0.044999999999999998]
# |                           (default: [0.037999999999999999, 0.0379999999999999
# |-DeltaRays               = True
# |-EnablePack              = True
# |-BeamBetaStar            = 2000.0  (default: 2000.0)
# |-Luminosity              = 1.23e+18  (default: 1.16e+18)
# |-CrossingRate            = 1.1245000000000002e-05  (default: 1.1245000000000
# |-DataPackingChecks       = True
# |-InteractionSize         = [0.032000000000000001, 0.032000000000000001, 38.1
# |                           (default: [0.027, 0.027, 38.199999999999996])
# |-Output                  = 'SIM'
# |-BeamMomentum            = 3500000.0  (default: 5000000.0)
# |-DataType                = ''
# |-DetectorSim             = {'VELO': ['Velo', 'PuVeto'], 'MUON': ['Muon'], '1
# |                           (default: {'VELO': ['Velo', 'PuVeto'], 'MUON': ['Mu
```

LHCb geometry in the simulation

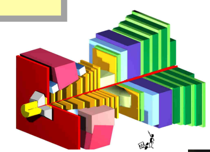
Gauss converts the LHCb geometry to the Geant4 description:



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'],…}
```

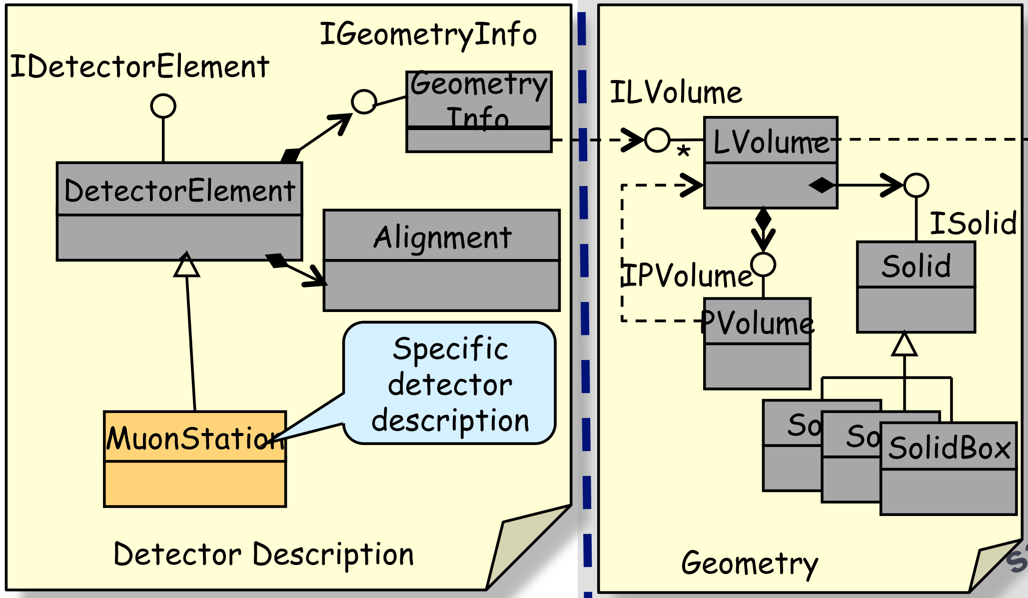
passed to appropriate GiGa component with pathway to find it in Transient Detector Store



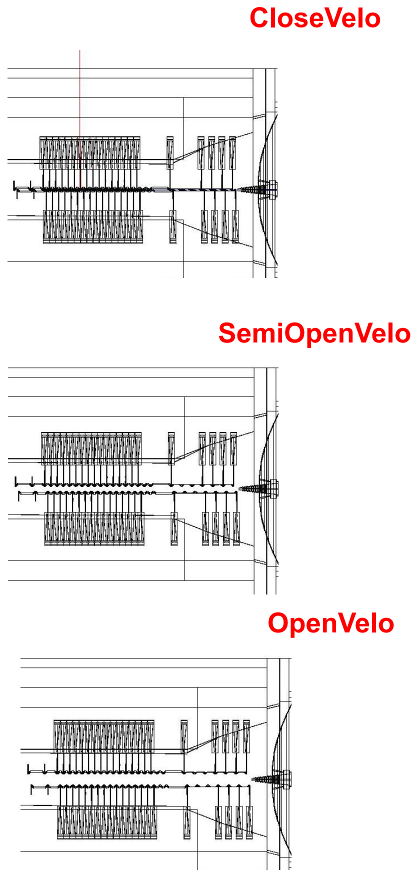
Geometry in the Simulation

info about misalignment

position as given in LHCb geo DB



same structure exists in G4: mapped one to one



GiGaDetectorElementCnv

only called when given to GiGa in list to simulate

GiGaLVolumeCnv

convert to G4 all its geometry tree

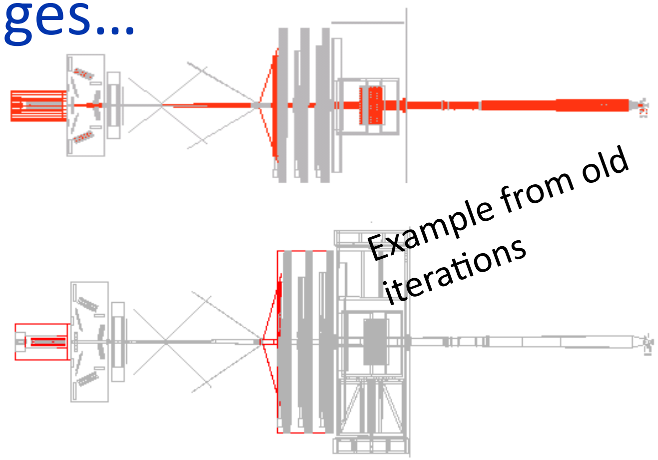
Limitation:

- 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 consistecy!

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

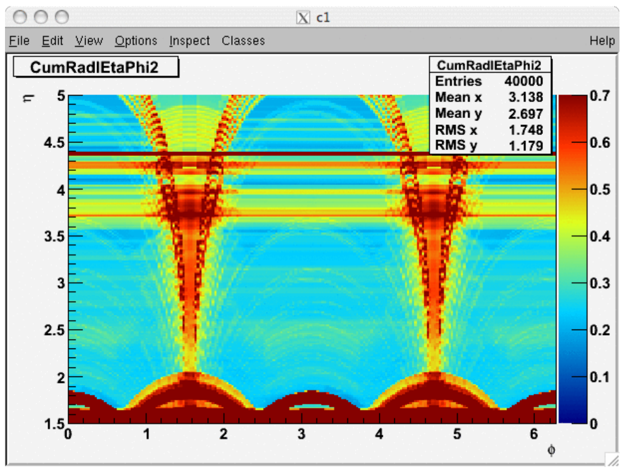
Must ensure when all mis-alignments are taken into account no overlaps are present

- 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 lenght 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)



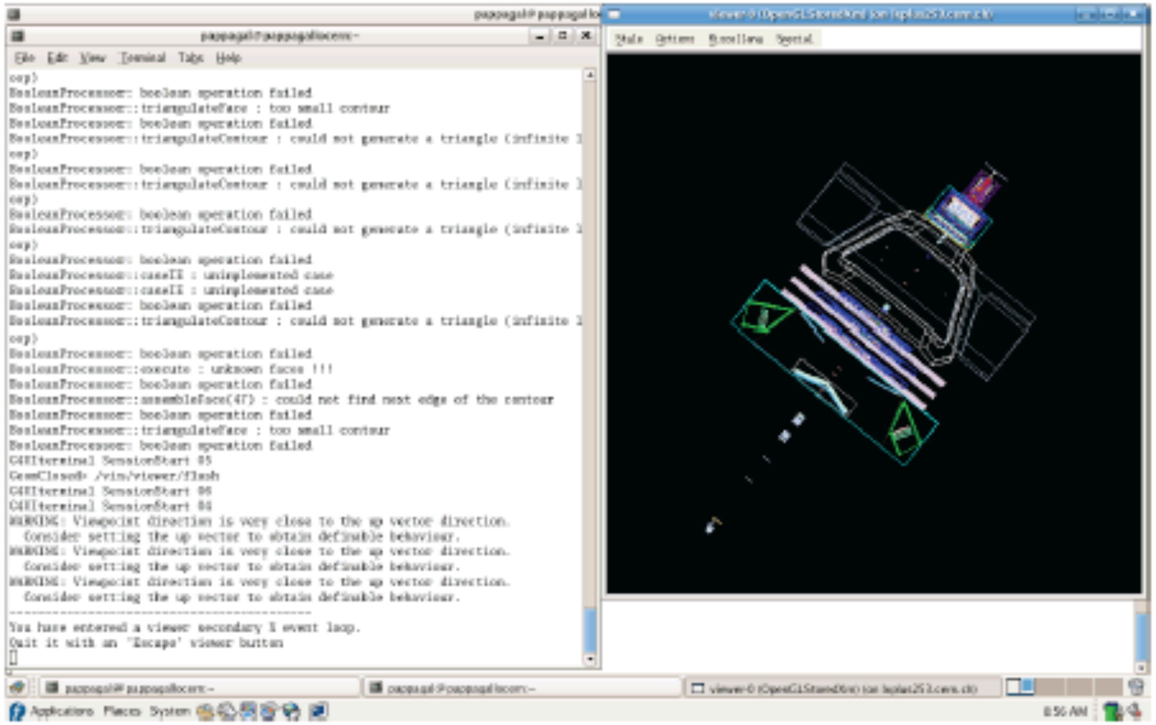
Geometry visualization

- 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.



Physics processes and lists

- 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

Physics Lists

- 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)

Physics processes and lists

- 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 instanciation 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

LHEP_EMV + LHCb HPDs processes

```
Gauss().PhysicsList = { ‘Em’: ‘Opt1’,  ‘Hadron’: ‘LHEP’, ‘General’: True, ‘LHCb’: True,
                        ‘Other’: False }
```

```
GiGa().ModularPL.PhysicsConstructors +=
  [ ‘GiGaExtPhysics<HadronPhysicsLHEP>/LHEPPhysics’ ]
GiGa().ModularPL.PhysicsConstructors +=
  [ ‘GiGaExtPhysics<G4HadronElasticPhysicsLHEP>/LHEPElastic’]
```

```
GiGa.ModularPL.PhysicsConstructors +=
  { "GiGaPhysConstructorOp"          } ;
GiGa.ModularPL.PhysicsConstructors +=
  { "GiGaPhysConstructorHpd"         } ;
```

Configuration of Physics lists to use

- Can also configure a builder via template extender
  - Templates customized to accommodate different constructors signatures
  - Could be automatised with common builders constructors with same wide-coverage signatures

```
Gauss().PhysicsList = { 'Em': 'LHCbTestNoCuts'…}
```

↓

```
GiGa().ModularPL.PhysicsConstructors +=
  [ 'GiGaExtPhysics<G4EmStandardPhysics_Opt1LHCb>/EmTest' ]
GiGa().ModularPL.EmTest.OutputLevel = DEBUG
GiGa().ModularPL.EmTest.ApplyCuts = False
```

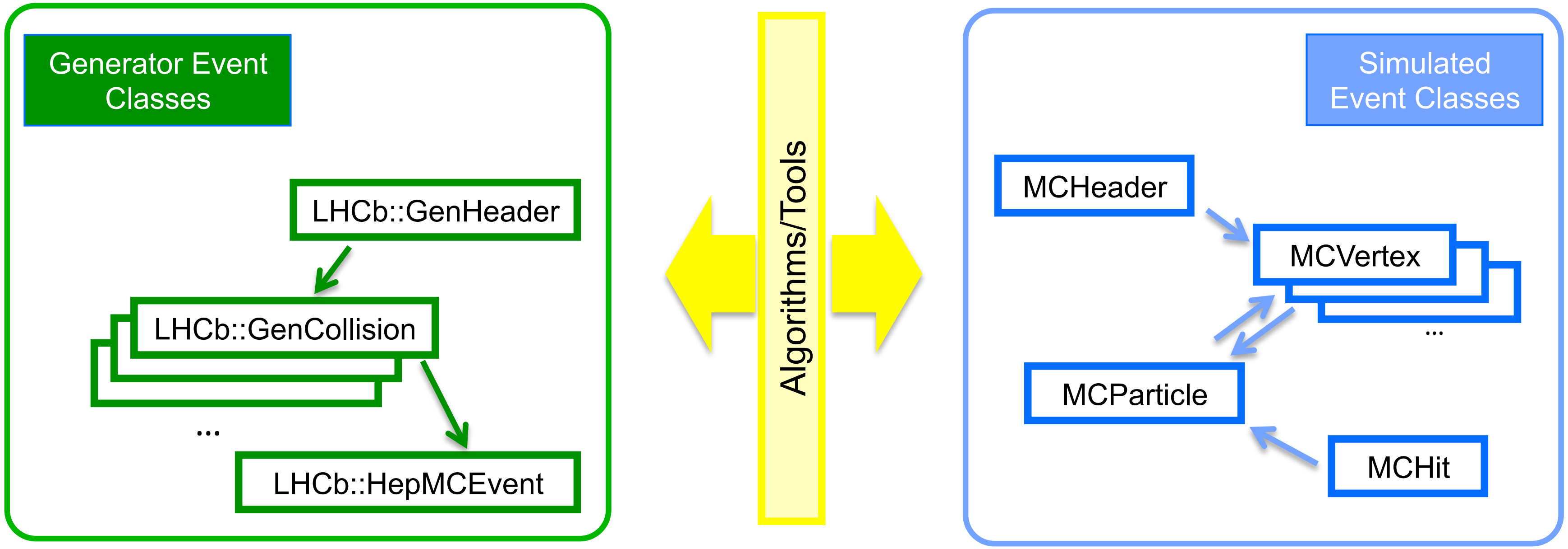
↓

In C++ classes transfer OutputLevel property to verbose and ApplyCuts to variable to be given in constructor

```
new G4EmStandardPhysics_Opt1LHCb(m_apply)
```

Production on distributed system steered via a different configuration file with only Gauss().PhysicsList specified

MC Event Model



- All classes inherit from LHCb DataObject and containers (keyed/contained object)
- **LHCb::HepMCEvent** is the LHCb wrapper around HepMC to provide general LHCb data object functionality: event record with all history, also used as exchange format within the generation
- History of particles traveling trough detector in dedicated LHCb event data classes **MCParticles and MCVertices** and their relationship

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

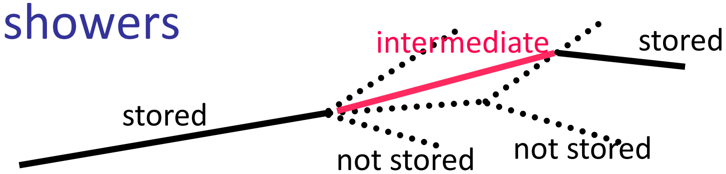
- Generator to Geant4 and MC truth
  - Pass only particles from generator to Geant4 which will interact with detector, that is to say particles with non-zero travel lenght.
  - All other particles are saved directly in MCParticle container, and the decay chains are restored at the end of the processing by Geant4.

- MCHistory (i.e. what happened during the tracking of particles) is essential to understand efficiencies and physics effects

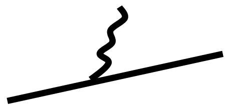
MC History

- Geant4 does not have a tree structure to keep history
  - interacting with tracks in G4 when a process occurs in StepAction is 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 Bhremsthralung and keep or not the split in algorithm transferring to MCParticles

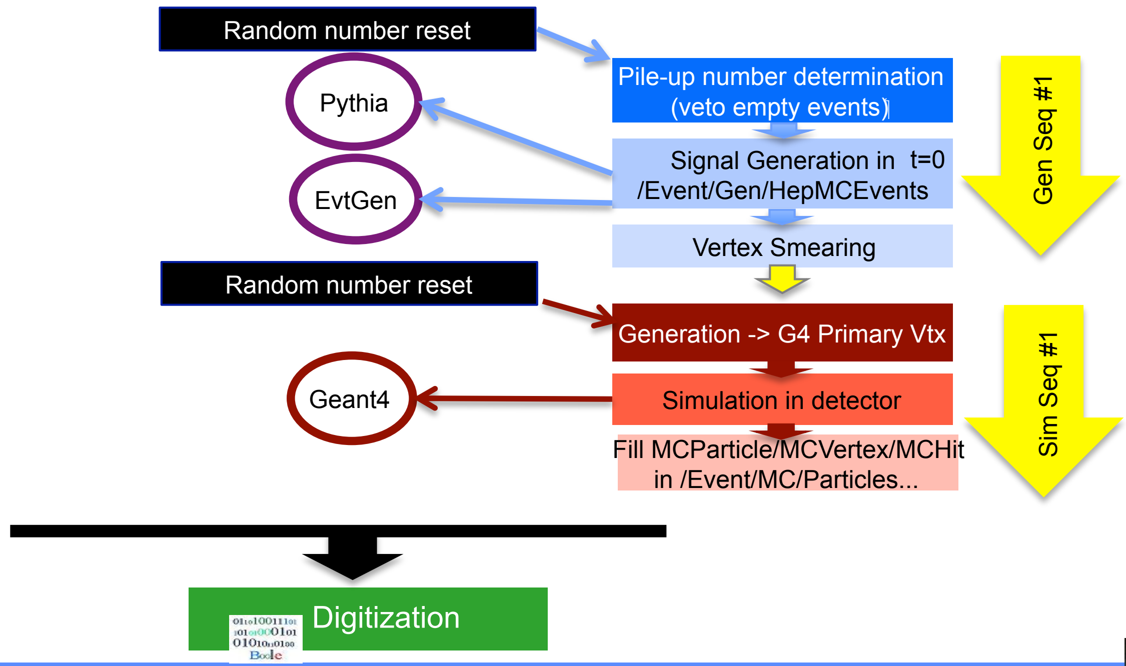


- Distinguish production mechanisms by having a global process identification in the vertices

Spill Over Events in Gauss

The treatment of Spill Over (SO) changed in 2009:

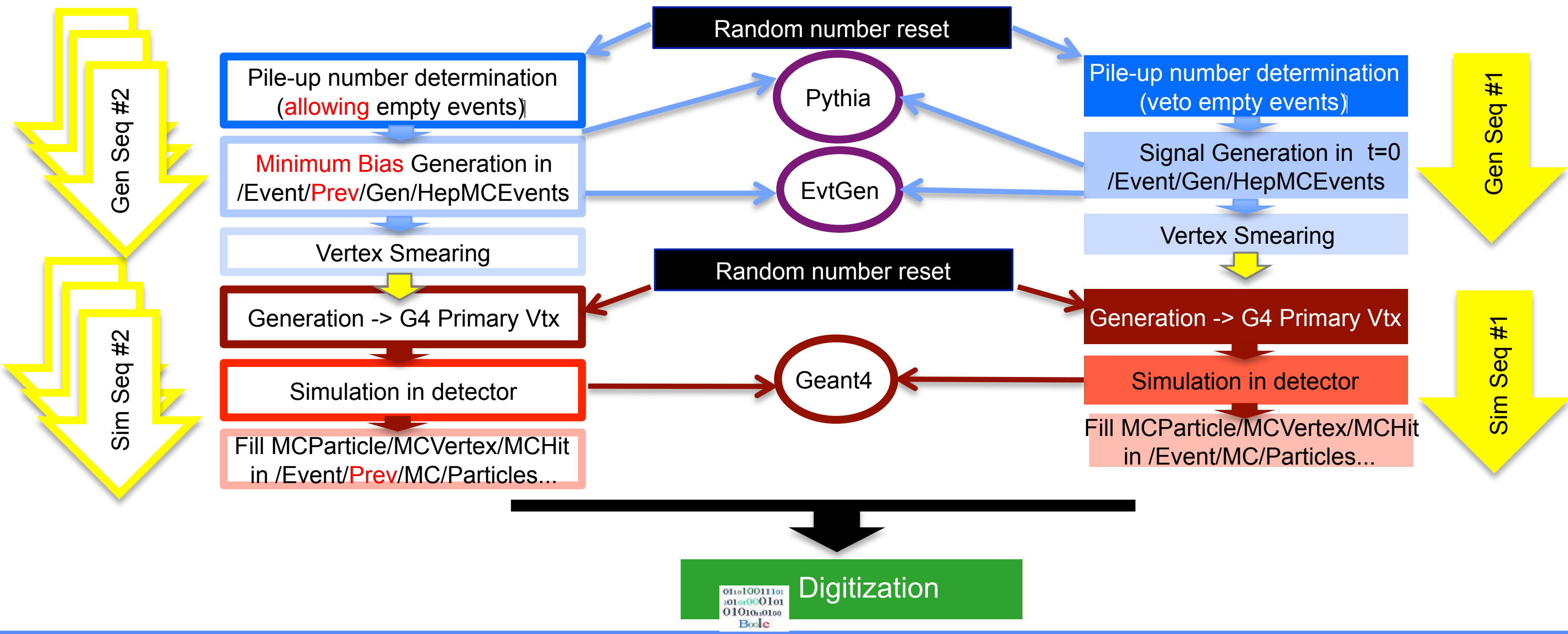
- previously treated in digitization application (Boole) using different input files (signal input file from Gauss and a minimum bias file merged)
- modified in order to generate spillover events in Gauss in a single file and a single job
- there is a single instance of Pythia, EvtGen, and Geant4 handling main event and SO events
- not a problem for Geant4 and EvtGen (decays): do not need to reconfigure the SO events



Spill Over Events in Gauss

The treatment of Spill Over (SO) changed in 2009:

- previously treated in digitization application (Boole) using different input files (signal input file from Gauss and a minimum bias file merged)
- modified in order to 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 SO events
- not a problem for Geant4 and EvtGen (decays): do not need to reconfigure the SO events



Simulation Monitoring

When adopting a new version of Gauss, a complete set of tests need to be performed to ensure the quality of the simulation:

- nightly build tests (QMTests)
- data quality
- geometry validation (overlaps checks, material budget, hit multiplicities,…) when a change is introduced
- simple physics validation
- full scale production tests

Gauss automatic software testing

- New releases are built with about ~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 (~10 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)

Geant4 and MC productions

- 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

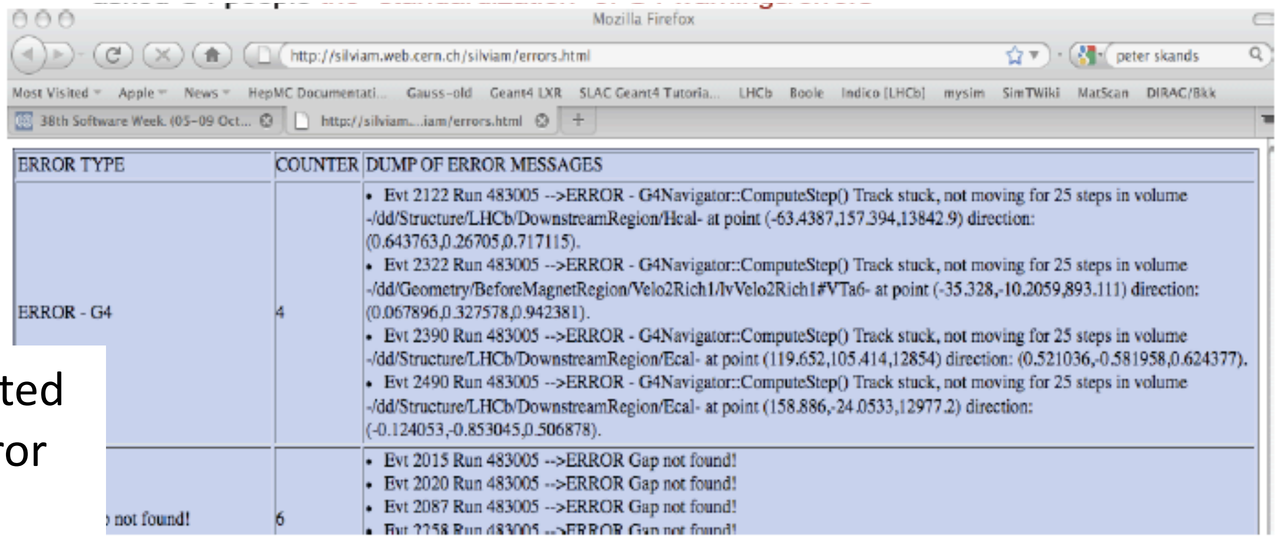
- Production of millions of events in a distributed environment require a stable application but with fast technical patches
  - Decouple physics from software issues as fixes for crashes and memory leaks or CPU time improvements

- Performance of the simulation may limit the amount of MC samples that can be produced within the computing resources of the experiment

Tracing problems in production

- Crashes and aborted events impact mass production
  - Protection introduced to protect from stuck particles and ping-pong effects

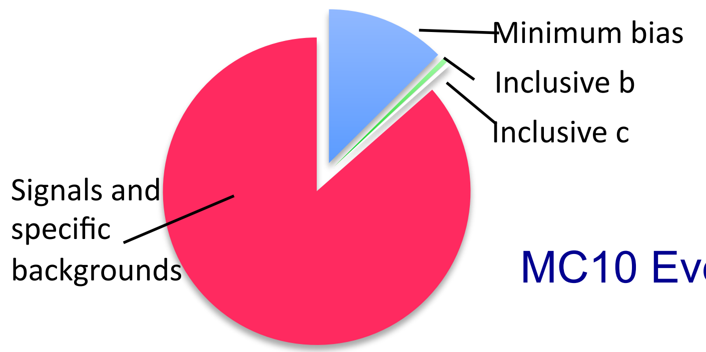
- Important to trace back reason of crashes and problems not leading to crashes (events or tracks aborted) during production
  - Impossible to look through ~50k jobs log files/day
  - Detect G4 errors at level of jobs during production and combine them for statistics
  - Retrieve full dump or error messages together with Run and Event Number to fully reproduce them and investigate them



Would be greatly facilitated by standardization of error messages

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	



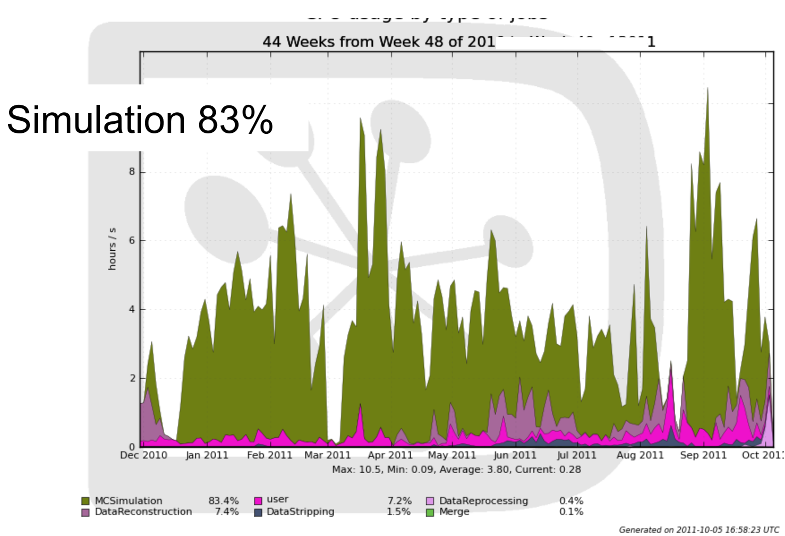
MC10 Event Types

Production on slc5 32-bit

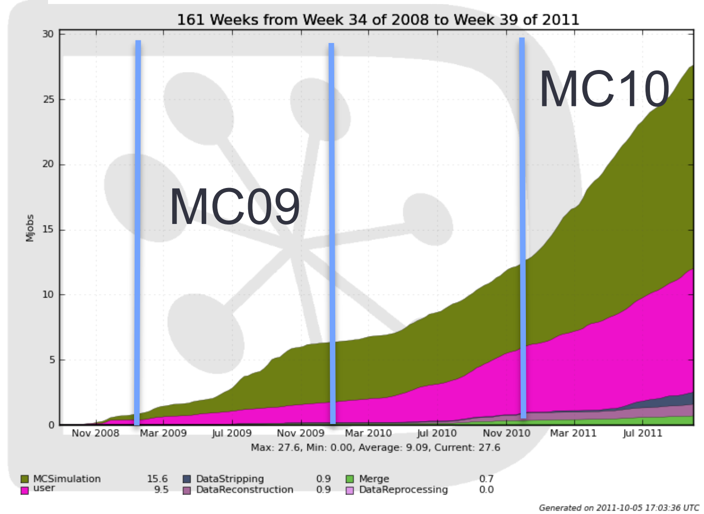
issue of jobs 'hanging' but not due to particles stuck on 64 bit, for now on a back-burner

Stable memory consumption

CPU usage by type of jobs



Cumulative jobs by type



Production and releases

- New model of continuous production over the year for MC samples for physics analysis starting

- Productions for analysis requires a well defined configuration with trigger and reconstruction consistent with data processing but at the same time continue development with the goal to improve the simulation and make it available as soon as possible
  - Two/three major releases of Gauss per year → Less coupling on G4 release dates but still do not change version more than once a year
    - May change physics lists and tuning in between
  - Time delay to adopt a new Geant4 release (in production for physics)
    - Simulation application may need to be adapted to new Geant4 version
    - Small tests allows to find big problems
  - Production-type tests necessary to find more subtle/rare problems and for physics and performance validation

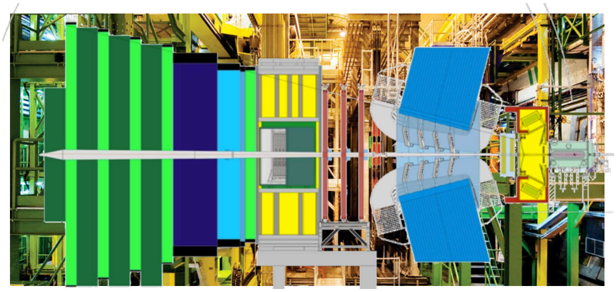
Summary

- Geant4 has been used in production and by users for their studies since 2004

- Technical choices determined by its encapsulation in the LHCb simulation package, Gauss and by functional requirements to provided flexibility and information for detector and physics studies

- Effort now shifting on data comparison to feed back on tuning and improvements

- Upgrade studies may provide new requirements
  - An occasion to review some of our technical choices

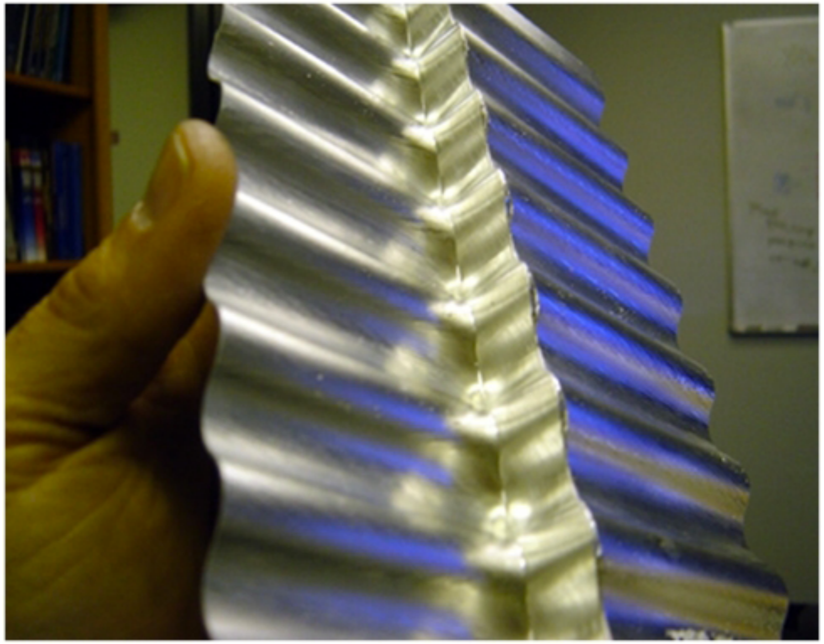


LPCC Workshop
“LHC Detectors Simulation: status, needs and prospects”
October 6th – 7th , 2011
CERN

Backup slides

Geometry in the simulation

- Feeding back corrections as from material studies with data as much as possible
- Investigating different solutions for upgrade with even more complex geometries



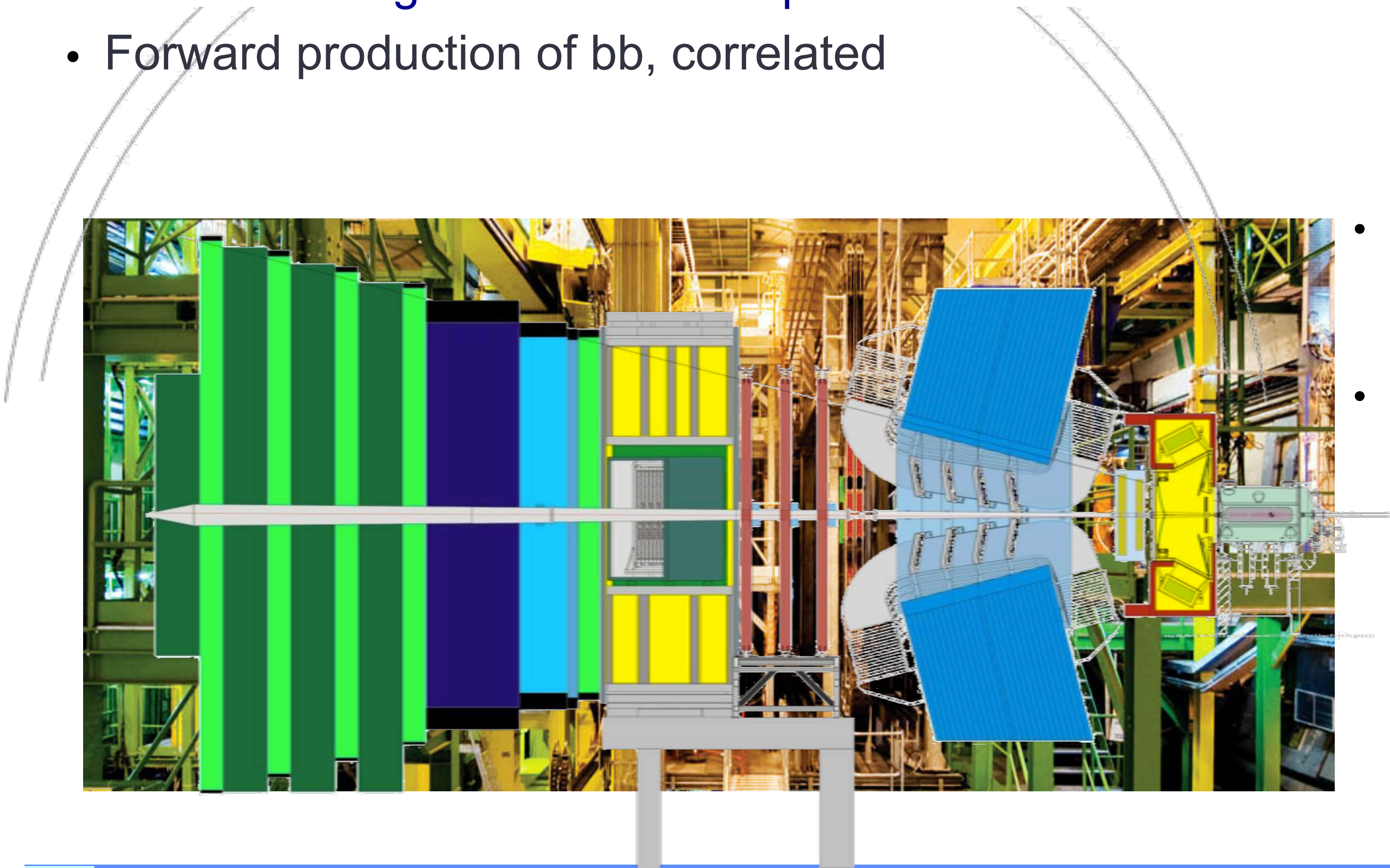
Wish list

- From Gauss
  - Review geometry conversion to Geant4 to allow for SIMCOND information to be fully used
  - May also need support for new type of volumes from Geant4
  - Make use of new features of Geant4 both on physics contents (e.g. new physics lists) and functional (e.g. process group identifiers)

- From Geant4
  - Physics builders common interfaces allowing configuration at construction or initialization time
  - Error and message standardization
  - Robustness and speed

LHCb

- Designed to make precision measurement of CP violation and other rare phenomena in the b system at the LHC
- Trigger and reconstruct many different B decay modes to make independent and complementary measurements
- LHCb is a single arm forward spectrometer
- Forward production of bb, correlated



- Amount of material in tracker area kept as low as possible ($0.6X_0$ up to RICH2)
- HCAL used mainly for trigger purpose

12 mrad < θ < 300 (250) mrad

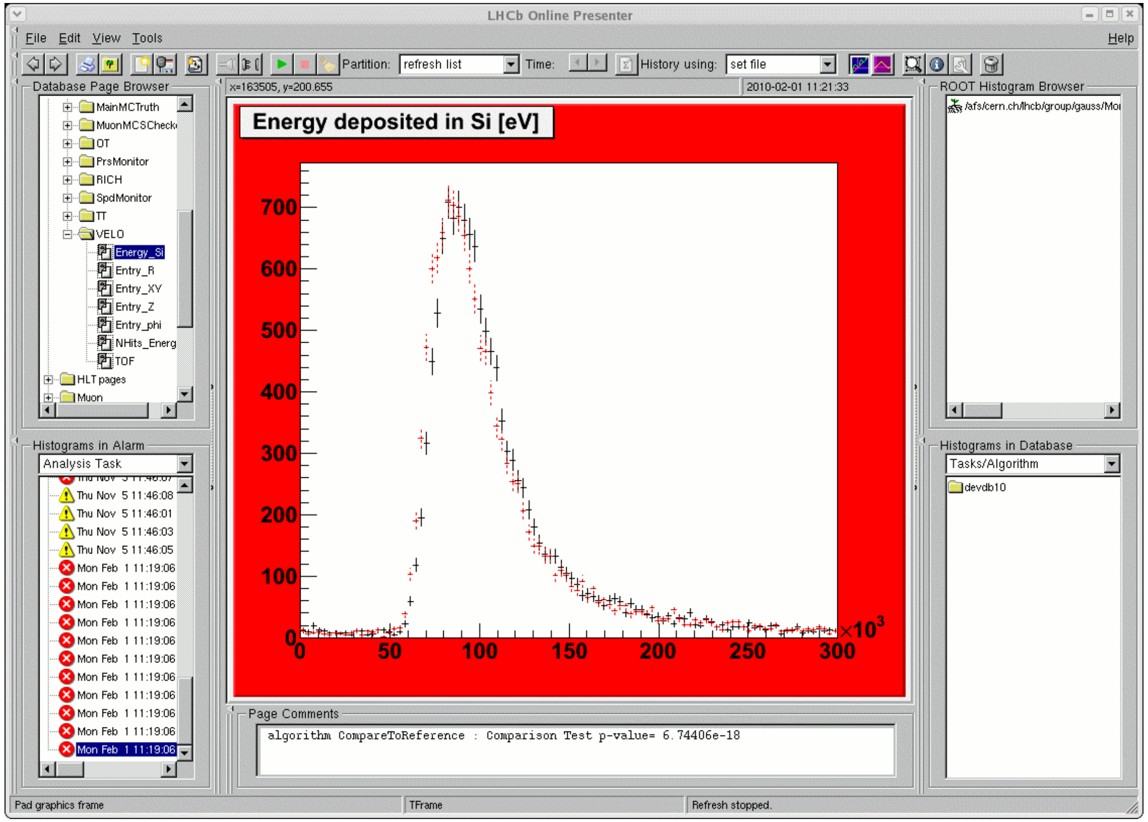
i.e. 2.0 < η < 4.9

For details Thursday N65-1

Data Quality

Proof of principle to integrate the complete set of Online Tools to monitor the Gauss output histograms:

- OnlineHistogramDB (sql-based) storing the display settings and configurations
- Online Presenter (GUI to display ROOT-based histos)
- Tools for Automatic Analysis can be run on the MC histos



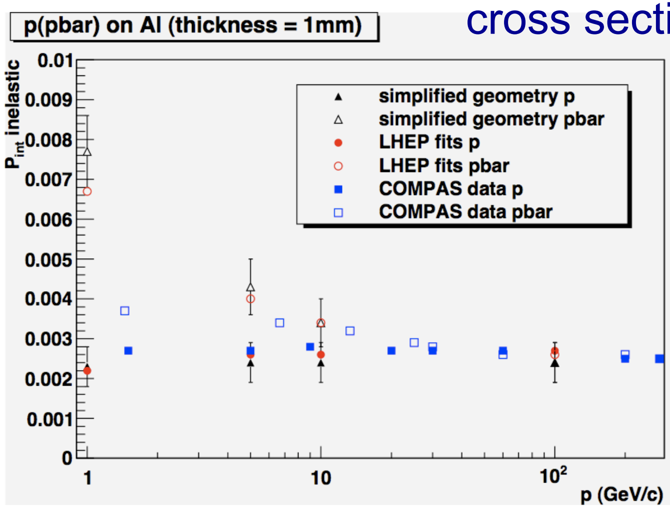
Geant4 Physics monitoring and validation

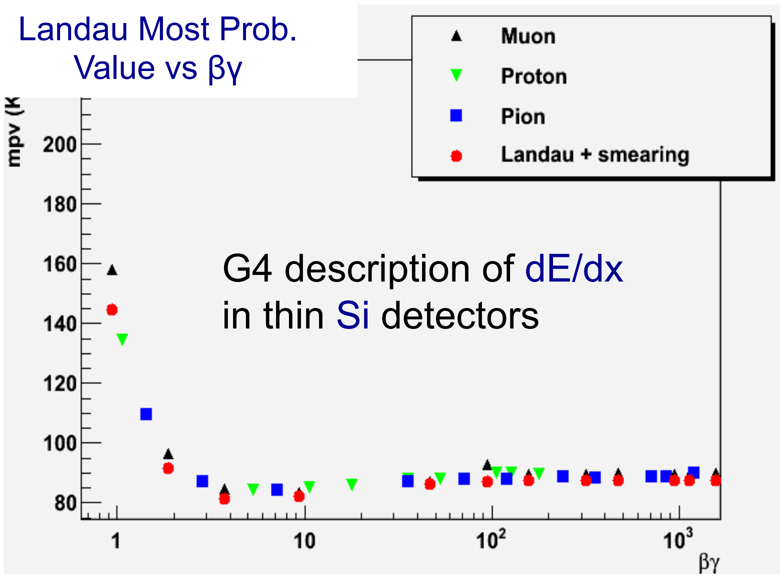
Every time a Geant4 version is changed:

- 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





G4 MCS