October 7, 2021

3rd Muon Community Meeting

Detector simulation software and computing at the Muon Collider

N. Bartosik [a]

on behalf of the
Muon Collider Detector and Physics Group

[a] INFN Torino *(Italy)*

Simulation overview

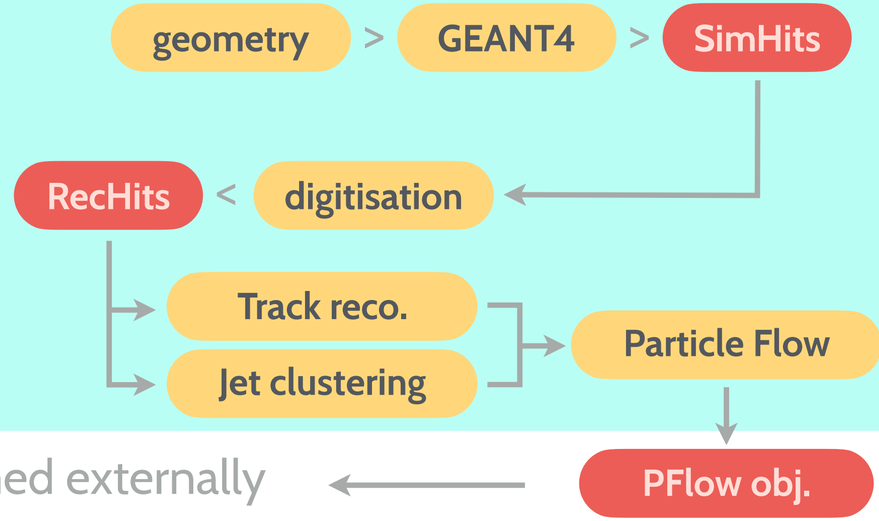
Main steps of a full-simulation study:

1. **generation of stable input particles:** ME + PS Pythia/Madgraph/Whizard + BIB FLUKA + FlukaLineBuilder

   Matrix Element + Parton Shower

   Beam Induced Background

2. **simulation of the detector response to the incoming particles** (DD4hep interface)
3. **simulation of detector effects** efficiency, electronics noise + thresholds, ...
4. **reconstruction of higher-level objects** photons, tracks, jets, particle identification



5. **higher-level analysis** ← can be performed externally

Detector simulation and event reconstruction handled within a single framework

↳ **inherited from the CLIC experiment:** comprehensive and modern workflow designed for e+e- colliders

Large overlap with the Key4HEP software stack: planning full transition in the future

Most of custom packages specific to the Muon Collider maintained in the public Muon Collider Software repository

Installation instructions available for CentOS 8 + Docker image for an easy and OS-independent local setup

Installation instructions: best for SW/algorithnm development

Docker image: best for data analysis with a fixed SW stack

ILCSoft framework

The main components of the ILCSoft framework:

1. LCIO **[L**inear **C**ollider **I/O]**
   Provides consistent storage of event data (**MCParticles**, **SimHits/RecHits**, **higher-level and custom objects**) using the ***.slcio** file format

2. Marlin **[M**odular **A**nalysis & **R**econstruction for the **Lin**ear collider**]**
   Collection of processors for isolated tasks that can be chained into an arbitrarily complex sequence of processes using XML configuration files
   - everything after hits simulated by GEANT4 is handled by processors within the Marlin framework: *digitization, track/jet reconstruction, b-tagging, Particle Flow, ...*

3. DD4hep **[D**etector **D**escription **for H**igh **E**nergy **P**hysics**]**
   Efficient and flexible detector geometry description with the interface to GEANT4 and Marlin

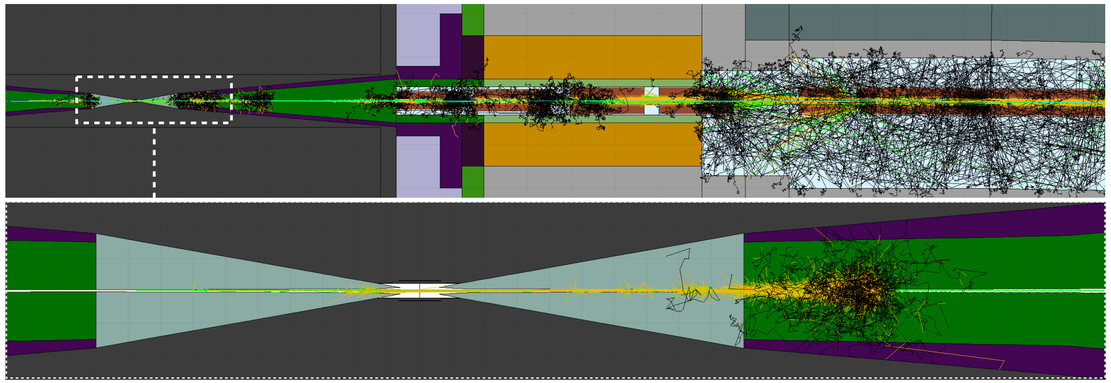
Beam Induced Background

The ultimate question we try to answer: what physics performance can we achieve?
↳ need to include BIB at every step of detector/MDI/reconstruction optimisations

Started with a single BIB sample generated by MAP for √s = 1.5 TeV (using MARS15)

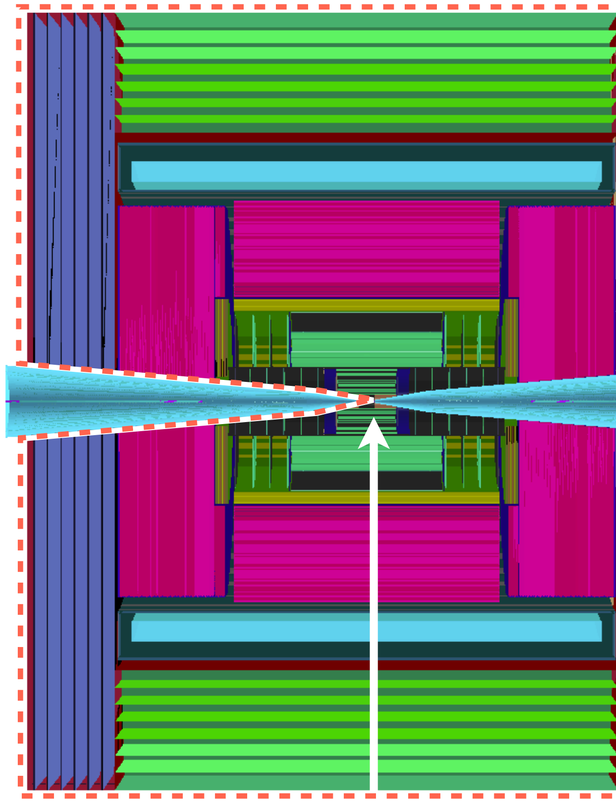
- list of stable BIB particles arriving to the detector volume
- collected at the outer surface of the detector + MDI ▶ GEANT4 detector simulation

New workflow in place based on FLUKA + FlukaLineBuilder



neutrons
photons
electrons/positrons
muons

- full control over the design of the accelerator lattice and MDI
- any beam energy can be studied
- coherent estimation of the radiation dose using a simplified detector geometry



Detector geometry based on the CLIC design

current version on GitHub

MARS15/FLUKA output converted to MCParticles in *.slcio format → native for the ILCSoft framework

Detector simulation workflow

Full simulated event obtained via three distinct stages:

GEANT4 simulation of Signal: straightforward and fast

GEANT4 simulation of BIB: ~10^8 particles/event

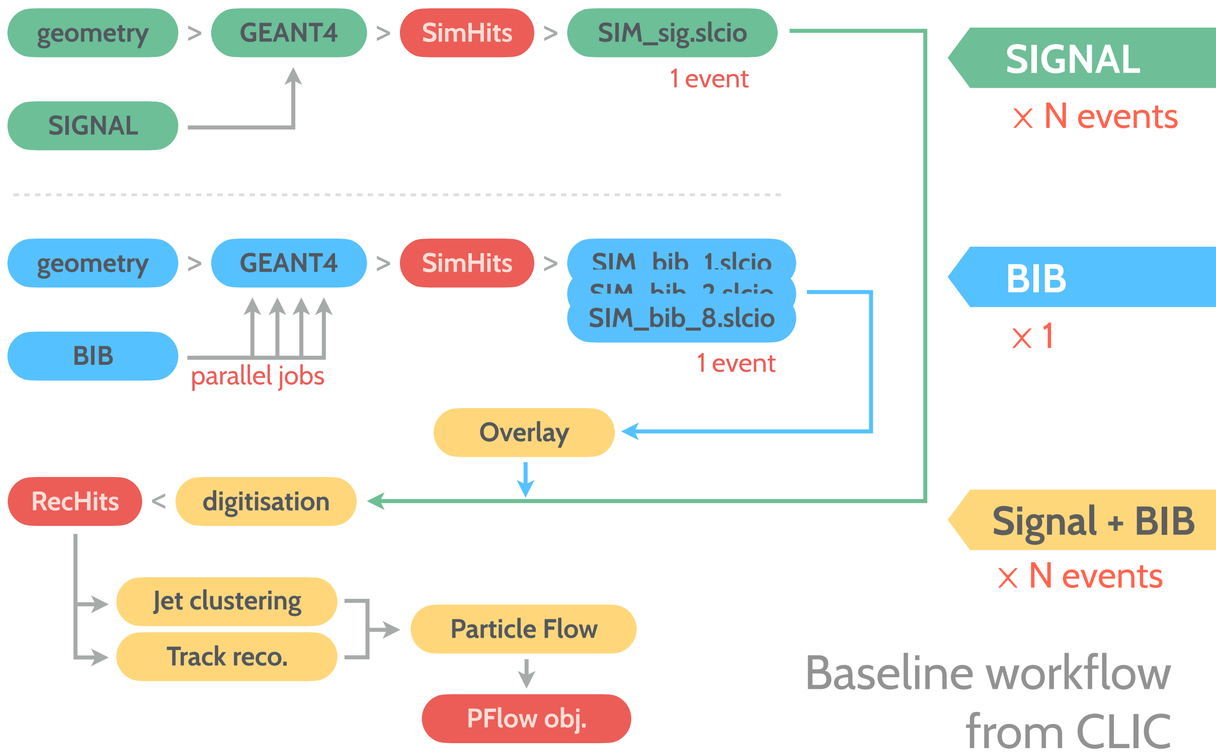
↳ sensitive to the # of input BIB particles

Overlay of BIB: performed in each event before digitisation

↳ sensitive to the # of BIB SimHits and merging logics

Reconstruction speed of higher-level objects strongly depends on the amount of input RecHits from BIB

- especially relevant for track reconstruction *(combinatorics)*
- BIB contribution has to be suppressed as early as possible



Baseline workflow from CLIC

BIB contribution creates tremendous amount of data → every step requires careful treatment of computing resources

DISK STORAGE | DISK I/O | CPU TIME | RAM USAGE | DISTRIBUTION

Properties of the BIB contribution

BIB has several characteristic features → crucial for its effective suppression

1. Predominantly very soft particles (p << 250 MeV) except for neutrons

fairly uniform distribution in the detector → no isolated signal-like deposits

↳ conceptually different from pile-up contributions at the LHC

2. Significant spread in time (few ns + long tails up to a few μs)

$\mu^+\mu^-$ collision time spread: 30ps (defined by the muon-beam properties)

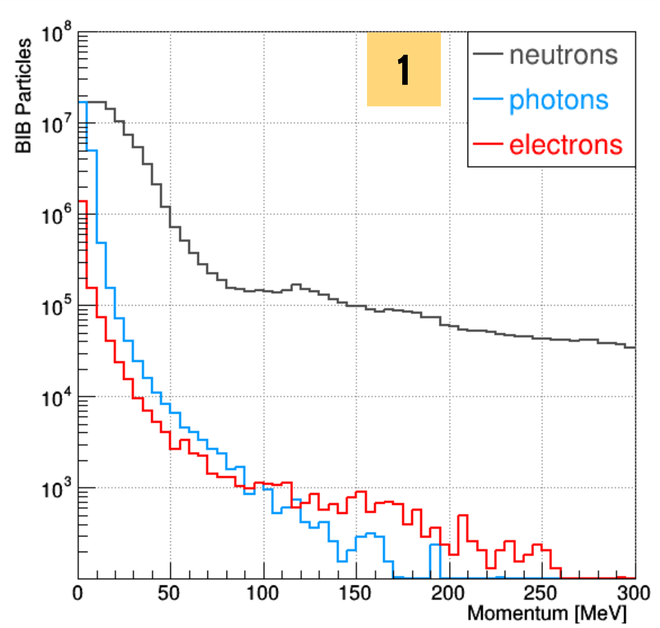
↳ strong handle on the BIB → requires state-of-the-art timing detectors

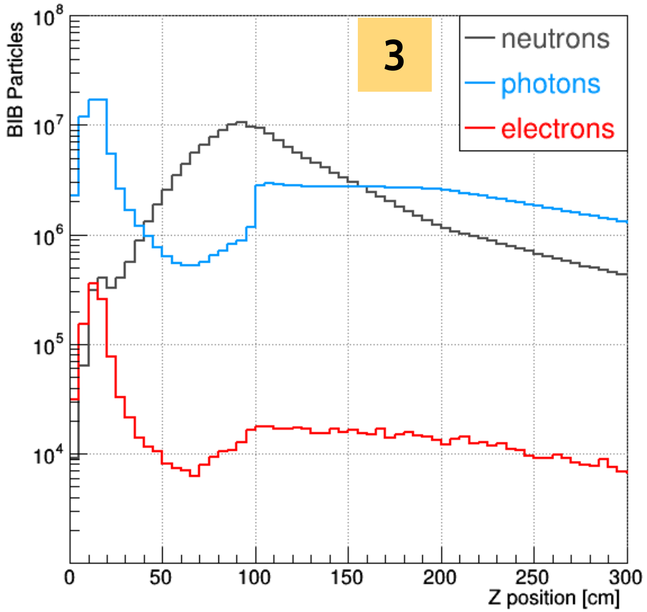
3. Large spread of the origin along the beam

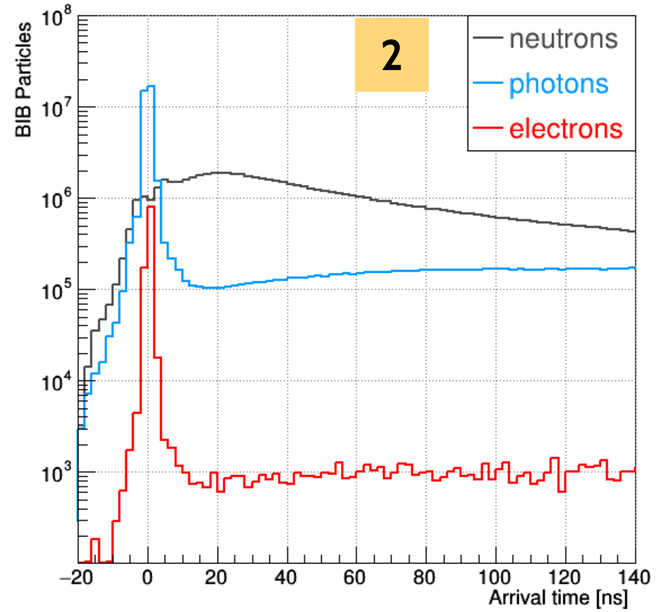
different azimuthal angle wrt the detector surface
+ affecting the time of flight to the detector

↳ relevant for position-sensitive detectors

An overview of the main optimisation steps and their impact presented in the recent paper







GEANT4 simulation

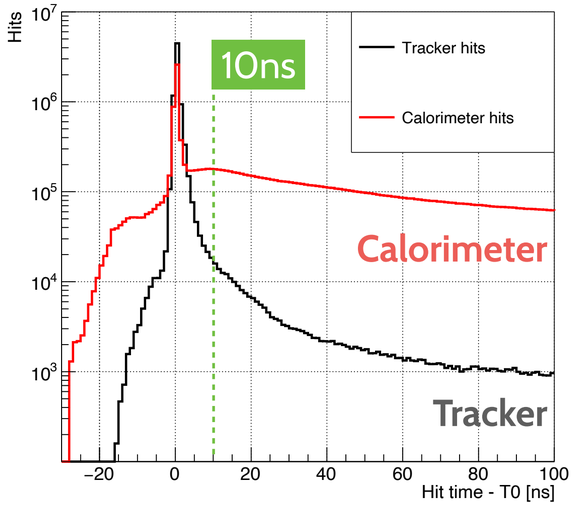
Not all of the ~10^8 BIB particles arriving to the detector are relevant for its performance in a real experiment
↳ **detectors have finite readout time windows → only a subset of particles relevant for the event reconstruction**

1. No GEANT4 simulation of particles arriving too late ×6 less CPU

hits at t > 10ns will be outside of the realistic readout time windows
↳ all particles with t > 25ns at the MDI surface are discarded *(accounting for TOF)*

2. No GEANT4 simulation of low-energy neutrons ×20 less CPU

high-precision neutron model required for accurate simulation: `QGSP_BERT_HP`
but they are slow → arrive to the detector with a significant delay
↳ neutrons with E_{kin} < 150 MeV can be safely excluded + faster model: `QGSP_BERT`

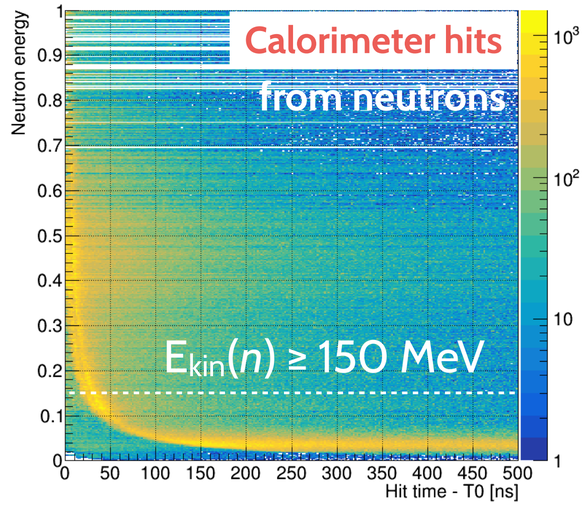


1

Improved GEANT4 simulation of a single BIB event from 127 days → 1 day
↳ ~10-100 reusable events can be generated in several days

3. Russian roulette sampling can be used for the most abundant particles *(<u>used in CMS</u>)*

- individual neutrons and photons from BIB are not reconstructable
- only combined energy deposits in calorimeters are relevant
  ↳ simulate a fraction (1/f) of particles with a weight (f) → less CPU/RAM/DISK



2

Digitisation logics

GEANT4 hits produced separately for Signal and BIB → merging + detector effects added during digitisation

↳ **two distinct classes of hits:** **CalorimeterHit** (ECAL, HCAL, Muon detector) + **TrackerHit** (Tracking detector)

1. Calorimeter hits: cell ID + E_{dep} + timestamp

large cells (0.5×0.5 - 3×3 cm) → manageable # of cells

↳ hits merged within a fixed readout time window *(0-10ns)*

2. TrackerHits: sensor ID + 2D position + time *and more*

small pixels (50×50 µm) to macro-pixels (0.05×10 mm)

↳ too many channels to treat them individually in GEANT4

2.1. Simple 3D smearing by $\sigma_U | \sigma_V | \sigma_t$ (30-60ps)

simple and fast (extension of the CLIC 2D smearing)

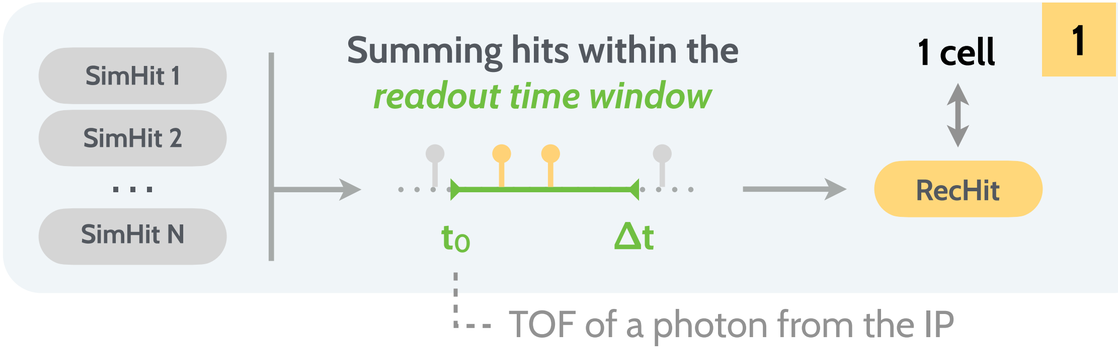
NO charge sharing, pile-up, electronics effects, etc.

2.2. Realistic simulation of sensor + readout-chip response

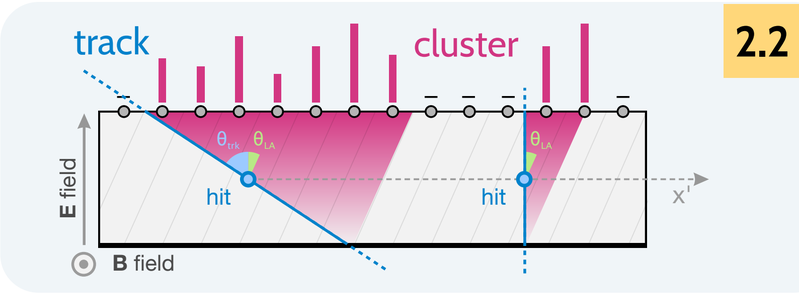
complex and slower (new development)

allows cluster-shape analysis for better BIB suppression

↳ more expensive digitisation → great savings in track reconstruction



realistic



Optimised digitization

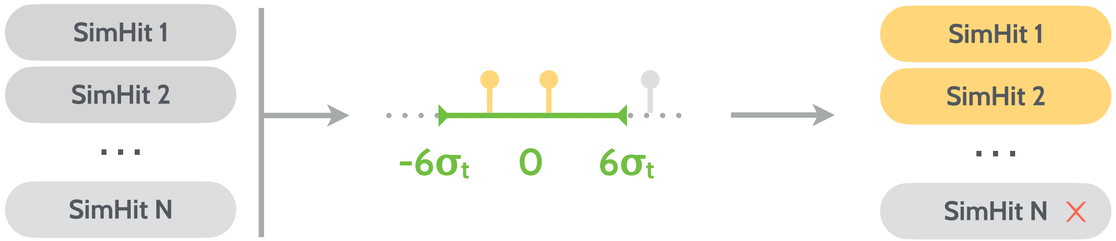
Tracker hits

Individual BIB particles have to be simulated

- **very small pixel size,** down to 50×50 μm
- **cluster-shape** → particle type/angle is important
- **precise timing** → pile-up effects are important

Hits created much later than the readout window will be discarded after digitisation anyway

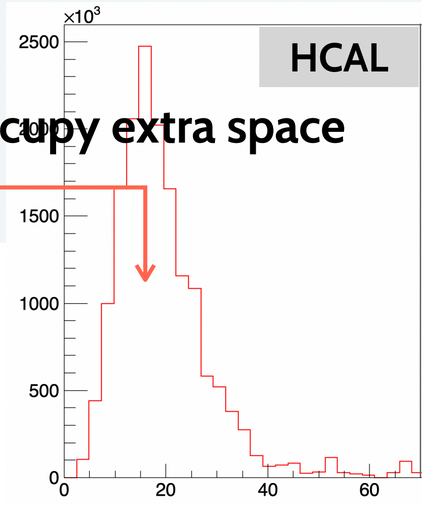
Extend time windows by extra ±3σ for SimHits
to account for the time smearing during digitization



Calorimeter hits

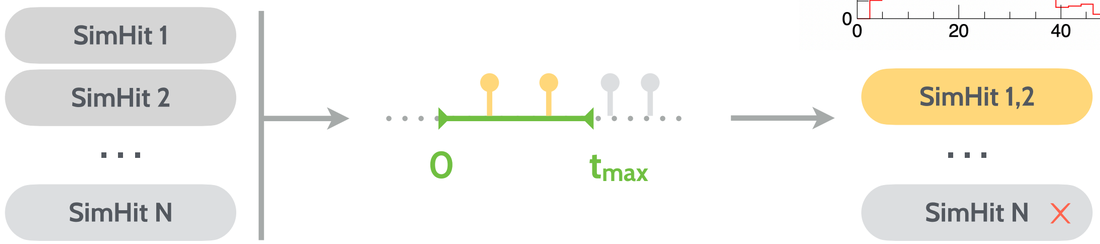
Only combined energy deposit in a single cell is relevant

- **large cell size,** at least 5×5 mm
- **less precise timing,** order of 100ps - 1ns
- **spatial distribution is fairly uniform**



Individual contributions in each cell occupy extra space
on average ×15 more than needed

Merge SimHits within a cell into one
before storing it to disk



Significantly reduced number of individual hits used as input in digitization processors

×10 less DISK/RAM

Tracking optimisation

Reconstruction of tracks suffers from large combinatorial background

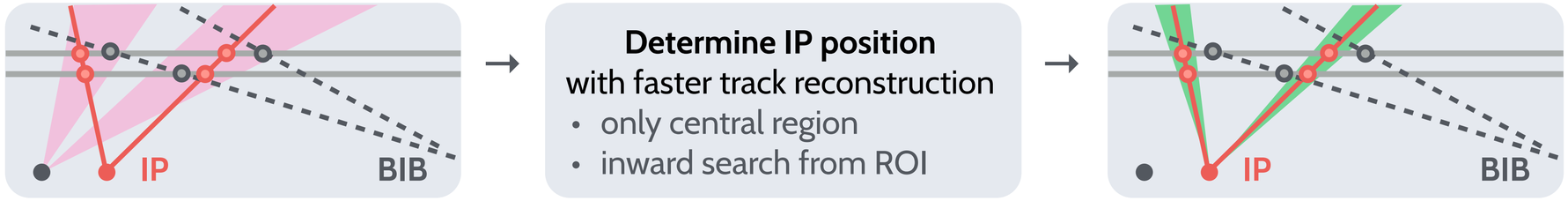
↳ **need suppression of BIB hits + efficient tracking strategies/algorithms**

1. Selection of hits in the narrow time window tailored to the sensor position

↳ limited by the time resolution + beamspot time spread + non-relativistic TOF

2. Selection of hit doublets aligned with the IP (double layers in the Vertex Detector)

↳ limited by the IP position resolution → requires multi-stage tracking strategy



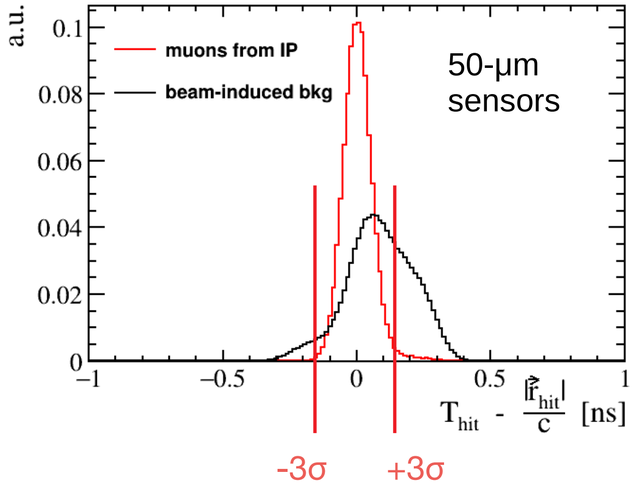
Impacts acceptance for displaced tracks affecting the b-tagging performance

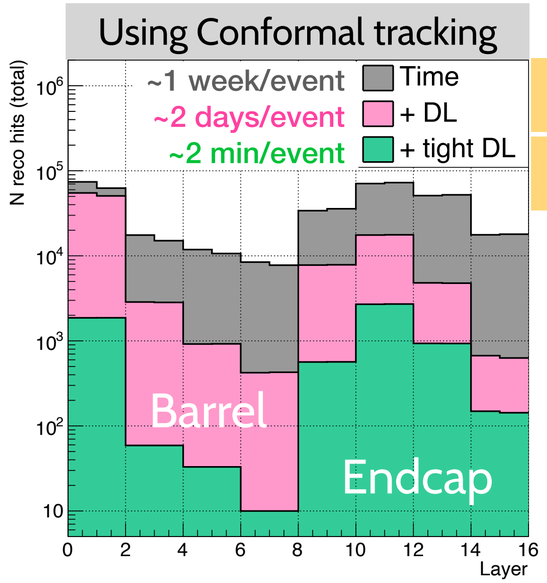
↳ has to be used with care, keeping the track topologies in mind

Trying to maintain each filtering stage as a standalone configurable processor

↳ multiple instances with independent selections for different regions or hit collections

- **FilterTimeHits:** selects input hits based on TOF for the assumed track velocity
- **FilterDoubleLayerHits:** selects hit pairs pointing to the centre of the detector





Latest developments

Several bigger software developments are currently ongoing

1. Realistic digitizer of the Si tracking sensors

- realistic transport of charge carriers and their distribution across the pixel grid
- accounting for pile-up contributions with clusters overlapping in space and time

2. Adoption of the ACTS tracking toolkit

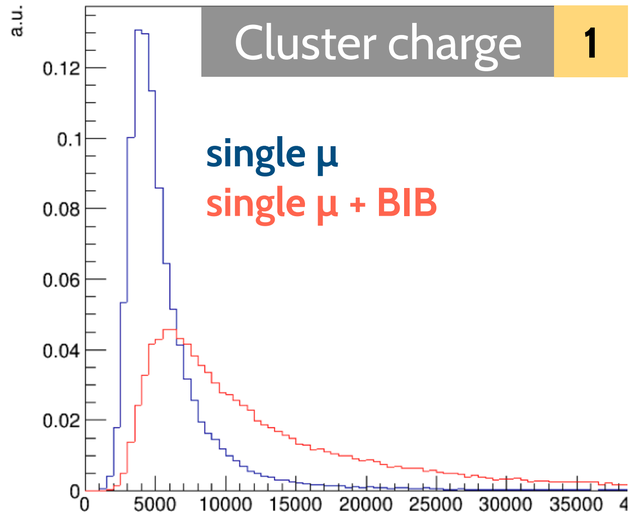
- highly optimised computational performance using vectorised calculations
- accurate accounting for the detector material budget
- interfacing with ILCSoft is successful → full integration into the framework is in progress

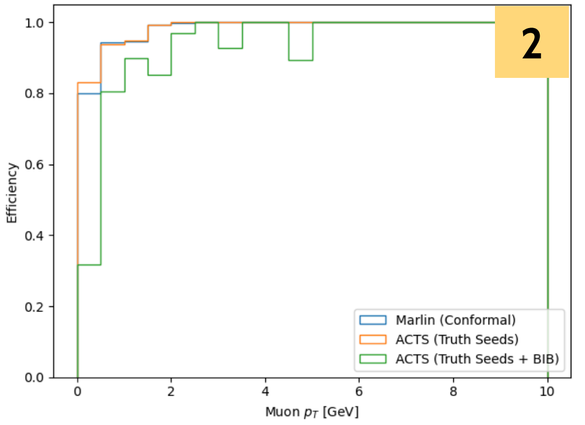
3. Parallelisation of Marlin processors

- would allow more efficient use of RAM in the most-CPU-demanding processes
- exact implementation strategy still under discussion

4. Automated procedure for the detector-level BIB characterisation

5. Alternative ECAL geometry based on the CRILIN design





+ many more developments: calorimeter BIB subtraction, muon-reconstruction, vertex reconstruction, b/c-jet tagging, ...

Summary

Muon Collider detector-simulation software is based on the solid ILCSoft framework
with centrally distributed releases

We now have full control over the BIB generation thanks to the FLUKA-based workflow

A lot of developments on top of the baseline CLIC version have been implemented
to provide sufficient computational performance in presence of BIB

Many ongoing developments are concentrated on improving the detector performance
and to further optimise the computing aspects of simulation studies