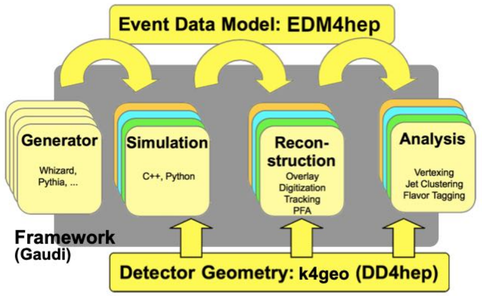
Software for FCC: simulation & reconstruction

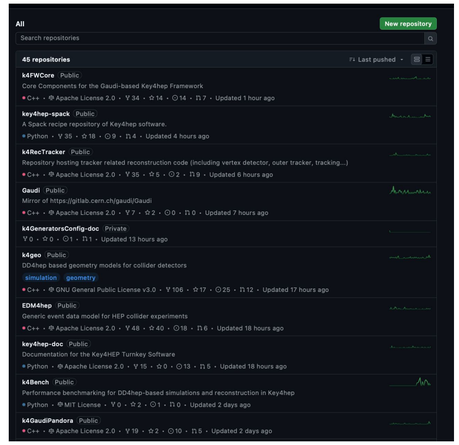
Giovanni Marchiori (APC Paris)

FCC-US workshop
9 September 2026

Overall infrastructure

- FCC simulation and reconstruction builds upon the **key4hep** software framework
  - and everything that comes with it: **edm4hep** event data model, **dd4hep** detector geometry description, **ROOT**-based I/O, **Geant4**-based detector simulation
- All (or most of) the code is based on the **Gaudi** infrastructure. Algorithms and tools written in C++, steered/run by python scripts
- Code is organised in (several) packages hosted on **GitHub**, e.g.
  - **k4geo**: geometry/segmentations
  - **k4RecTracker**: tracker reconstruction code
  - **k4RecCalorimeter**: calorimeter reconstruction code
  - **FCC-config**: steering files for digitisation/reconstruction
  - **Detector-specific recipes** (e.g. **ALLEGRO**: scripts to produce inputs for digitisation/reconstruction)





- Large code-base also inherited/available from LC, some algorithms already migrated, other reusable via **k4<->Marlin "wrappers"**
- **Nightlies** built every day; **releases** created from tagged versions of the packages from time to time
  - Setup keystack for nightly with `source /cvmfs/sw-nightlies.hsf.org/key4hep/setup.sh`
- Development still ongoing at fast pace - releases tend to be outdated rapidly
- More details in David's talk at this workshop as well as in Juan's talk at the FCC week

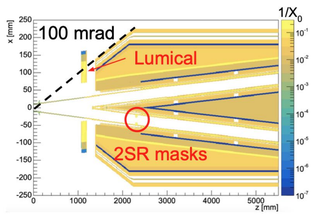
Detector geometry

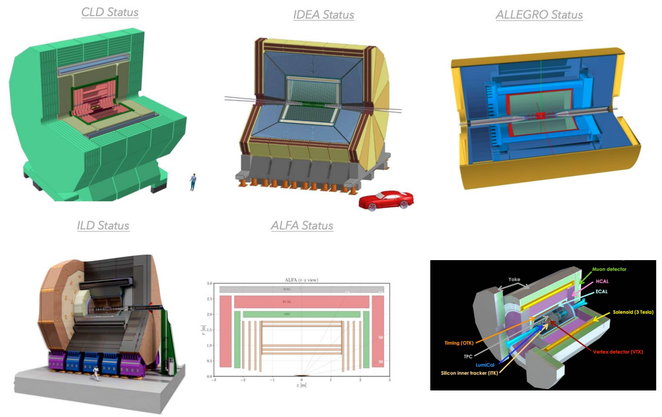
- **Geometries** implemented in **DD4hep** and available in **k4geo**
  - C++ detector builders (aka drivers) configured via xml files to set the free parameters (geometry, materials, readout)
  - Enables sub-systems plug and play and flexible geometries
  - Target simplest geometry possible which captures the dominant physics effects (e.g. services as simple native shape with equivalent material budget)
  - Try to write detector builders that automatically adapt upon external envelope changes

```
<constant name="VTX_r_min" value="VTXIB_r_min_layer-VTXIB_r_min_clearance"/>
<constant name="VTX_r_max" value="VTXOB_r_max_layer+VTXOB_rmax_clearance"/>
<constant name="VTX_z_max" value="VTXD_z_max+4.5*cm"/>
<!-- End of VTX parameters -->

<!--  Drift Chamber parameters    -->
<!-- Changing the radius is not enough to resize the detector, please contact
<constant name="DCH_inner_cyl_R_total"          value=" 349.8  * mm"  />
<constant name="DCH_outer_cyl_R_total"          value=" 2015.   * mm"  />
<constant name="DCH_half_length_total"          value=" 2250.   * mm"  />
```

- Simple **versioning** system implemented in k4geo via subdirectory naming scheme: `<detector>_o<X>_v<YY>` where
  - X : major detector change e.g. sub detector technology
  - YY : minor change e.g. geometry parameters / materials
- 6 **detector concepts**: geometries implemented (in k4geo/FCCee) for ILD_FCCee, IDEA, ALLEGRO, CLD; work ongoing on ALFA; AGORA to be ported from CEPC repository
  - **MDI** geometry also implemented
  - More details in Charles/Vhako talk later and Joshua's talk at FCC week



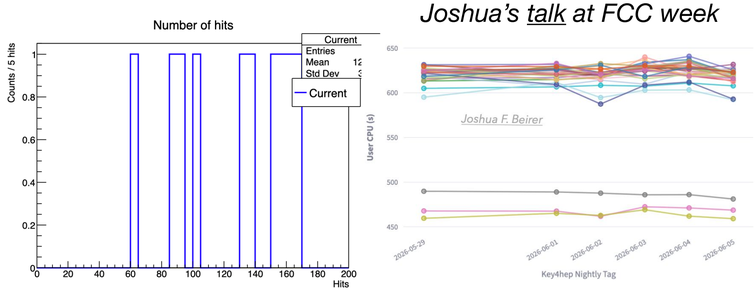


- Many degrees of freedom (geometry, materials, ..) not frozen - under optimisation

Simulation

- Run with **ddsim** passing options via CLI or steering file:
  `ddsim --compactFile $K4GEO/FCCee/<detector>/compact/<detector_oX_vYY>/<detector_oX_vYY.xml> <options>`
- Uses **Geant4** under the hood
- **Event generation** can be performed within `ddsim` using a particle gun, or externally (eg w/ Pythia), saved to a file and passed to `ddsim` via `-I <inputFiles>`, multiple formats eg `.stdhep, .hepmc, .hepmc3` accepted
- **ROOT** file is produced in output, in **EDM4hep** data format - contains `events` TTree with
  - *Event* info (runNumber, eventNumber, weights, ..): `EventHeader`
  - *MC truth* particles (kinematics, parent/daugther links): `MCParticles`
  - *Tracker* hits
    - typically 1 hit/G4 energy deposition => `cellID, eDep, time, position, momentum` of particle
    - link to `MCParticle` that produced the hit (e.g. `_VertexBarrelCollection_particle`)
  - *Calo* hits
    - All G4 deposits within 1 readout cell summed together into a `SimCaloHit`, storing `cellID, energy, position`, and links to G4 energy deposits in separate collection (holding PDG, energy, time, step 3D position, step length, and link to `MCParticle`)
    - Readouts defined via “segmentation” classes in k4geo
- PR to make ddsim **multithreaded** merged few weeks ago
- Tools for **validation and regression:**
  - **Physics-based simulation + reconstruction validation:** https://key4hep-validation.web.cern.ch
  - **Memory/CPU monitoring of simulation:** https://k4bench-dashboard.app.cern.ch

Joshua’s talk at FCC week



Digitisation and reconstruction overview

- Run with **k4run,** using **python steering scripts** to configure the algorithms:
  `k4run $FCCCONFIG/FCCee/FullSim/<detector>/<detector_oX_vYY>/<steering_script.py> <options>`

- Two types of algorithms run

  - **Digitisers**: convert hits into digitised cells (mimic detector readout chain)

  - **Reconstruction algorithms**: use digitised cells to find high-level objects (tracks, clusters) and potentially refine them (apply some calibrations, perform some particle identification, …)

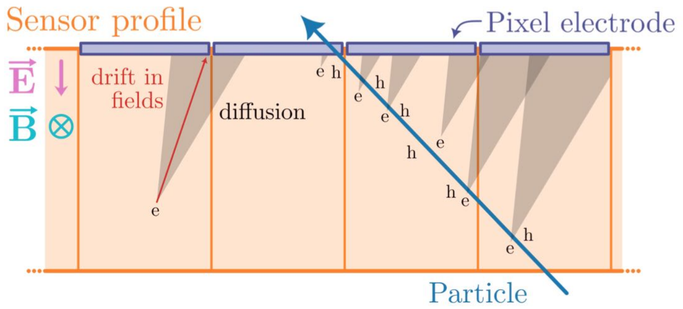
    - Some higher-level reconstruction (e.g. jet clustering and flavour tagging; global event reconstruction) possibly deferred to separate step in analysis chain, and not covered here (see e.g. dedicated talk by Dolores on p-flow reconstruction)

- Output —> **ROOT** file in **EDM4hep** data format, contains TTree `"events"` with digitised hits (aka cells for calo-type detectors), input hits, particles, higher-level objects (e.g. calo cell clusters), links between objects

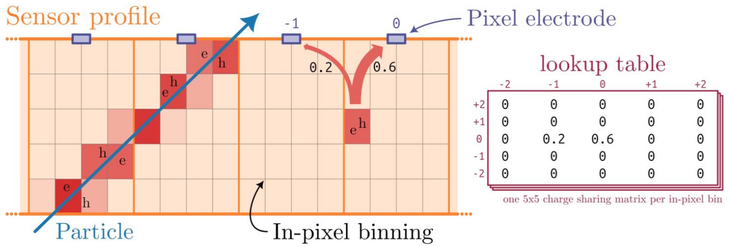
  - Output is **configurable** - collections can be dropped to reduce file size (`io_svc.outputCommands = ["keep *","drop …"]`)

Available digitisers - tracker (planar sensors)

- **Basic**: `DDPlanarDigi`: **Gaussian smearing** of position (local x and y) and time based on expected resolution
- **More realistic**, with better modelling of spatial effects
  - `VTXdigitizerDetailed`: **projection-based**, projects deposited charges to sensor surface given Lorentz angle, smears according to expected thermal diffusion.
    - Can apply per-pixel noise and threshold to charge collected.
    - In the nightlies, available for generic studies



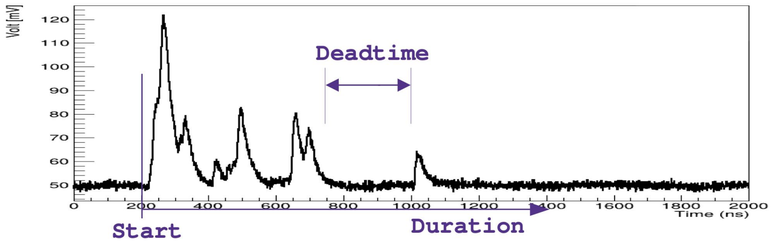
  - `VTXdigi_Modular`: l**ookup-table-based**, uses charge transport maps from R&D studies (E-field from TCAD + charge transport from AllPix²) to encode charge sharing behaviour.
    - Each pixel is divided into a finer 2D grid, for each bin a LUT of charge sharing to this and neighbouring pixel is used.
    - PR open, for expert-level studies (compare sensor designs)

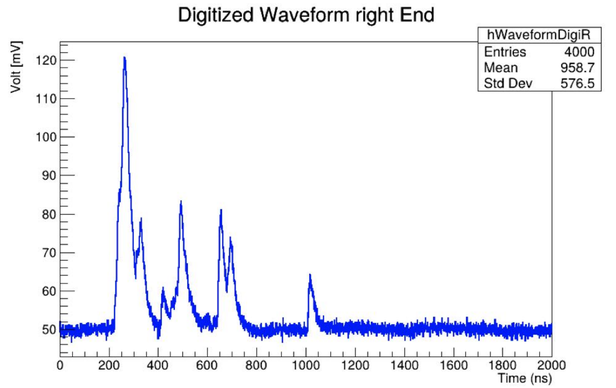


- More details in Jonas' talk at FCC week

Available digitisers - tracker (gaseous detectors)

- **Drift chamber**:
  - `DCHdigi_v02`: creates one digi per (hit) wire, grouping together hits within configurable readout window. **Smears position and time. Accounts for drift and signal propagation time. Gives N(ionisation clusters) per cell** (->dN_{cl}/dx, for PID) from Garfield parameterisation — as function of gas and particle's βγ — taken from Delphes (from hits step length and Poisson sampling, using MCParticles associated to hit)
    - More details in Andreas' talk at FCC physics workshop
  - `DCHdigi_v03`: under development (open PR), **several refinements**: tabulated drift time/ distance relation; reweigh Geant4 hits to make distance distribution exponential with proper ionisation length; can use dN/dx from testbeam; mimic production of primary electrons + drift/ diffusion, gain (avalanche), propagation time/attenuation length/rise+fall time, add noise to build analog waveform; emulate ADC **to get digitised waveform for each wire**
    - More details in Muhammad's talk at FCC Full-Sim meeting
  - Written with DCH in mind, but **being adapted to straw-tube tracker** (DCHdigi_v02 -> WireTrackerDigi_v01, open PR)



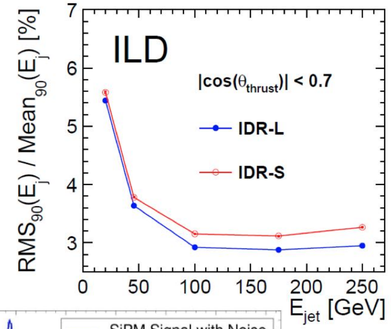


- **TPC**:
  - `DDTPCDigiProcessor`: written for ILD (in Marlin), **for pad readout** (6mm row-based) - **but used also for pixel readout** (link). G4 deposits smeared and merged based on TB results, with parameterised dependence on drift length and angle. Approach **developed for bunch structure at LC, need to investigate impact of vastly more ions in irregular patters at CC** on e.g. point resolution. dE/dx using 70% truncated mean of hit energies of reconstructed track

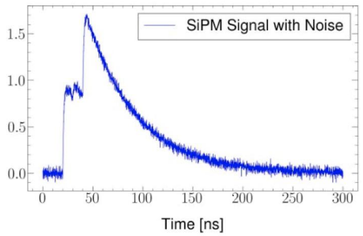
Available digitisers - calorimeters, PID

- `CreatePositionedCaloCells`: very simple, currently used for **ALLEGRO sampling calorimeters (LAr/Pb ECAL, steel/tile HCAL)**.
  - Sums G4 energy deposits with same cell ID, correct for pre-computed per-layer sampling fraction. Can emulate noise and x-talk (based on expected noise level and x-talk coefficients) and filter cells with E (or |E|)<N*noise

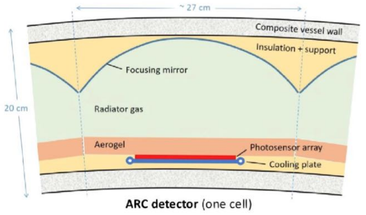
ILD IDR 2020

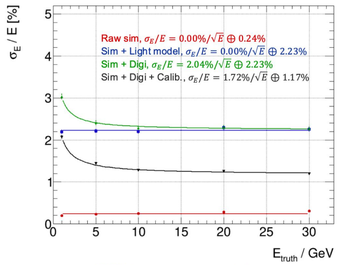


- `DDCaloDigi, DigitiserSiliconMip, DigitiserScintPpdNpe`: used by **CLD/ILD/MAIA**, ported/ being ported (open PR) from Marlin, for digitisation of Si-W and scintillator-SiPM sampling calorimeters
  - Detailed cell-level calibrations/corrections, timing (signal propagation, readout window, time resolution, ..), dead cells, miscalibrations, electronic noise..



- `SimulateSiPMWithContrib`: for digitation of hits in IDEA dual-readout calorimeters (crystal ECAL, fibre HCAL) to model the SiPM response, including multiple effects (dark count rate, crosstalk, afterpulsing, cell recovery time, signal rise/fall times) -> more details
  - OK for fibre calorimeter, work ongoing to have digitisation working for ECAL too (calculate time of arrival of p.e. at crystal SiPMs)
- `DDSimpleMuonDigi`: for muon tagger (ILD/CLD). Applies global energy correction factor, keeps hits with E>threshold, computes hit time as instant when cumulative sum of G4 deposits' energies crosses the energy threshold.
- More digitisers under development for **Grainita** and for **ALLEGRO LAr calo**
- Work needed for digitisation of **ARC** (array of RICH cells)

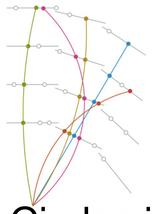




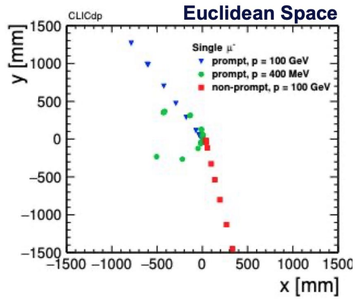
W/ light response model

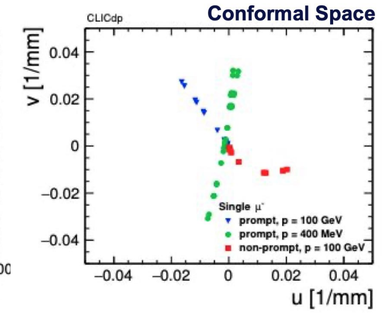
Reconstruction - tracking 1/2

- Two-step problem: track finding (pattern recognition), track fitting (refining track parameter, estimating uncertainties)
- **Pattern recognition**:



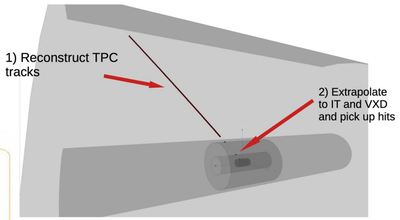
  - **Without ML**:
    - **Conformal mapping**: Circles in xy plane become straight lines in conformal space
      - Available in key4hep (`ConformalTracking`), does not work for drift chamber hits (drift isochrones require alternative pattern recognition) (more details)
      - Used by CLD (full-Si detector), usable for ALFA and for vertex+Si-wrapper in e.g. ALLEGRO, IDEA..



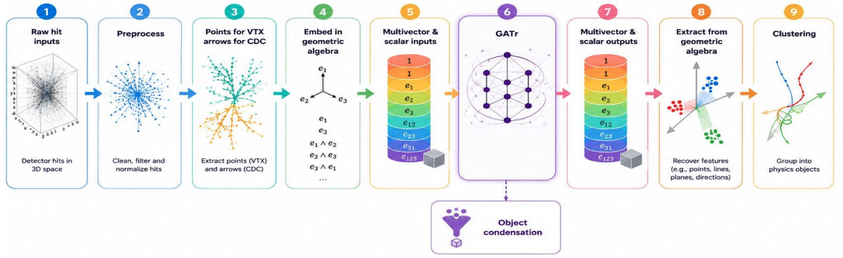


    - **ACTS**: experiment-independent toolkit for track reconstruction in HEP
      - Available in key4hep (`k4ActsTracking`), same limitations as conformal tracking, work ongoing to develop dedicated pattern matching for drift chambers (Hough-transform) (more details)

    - **Hybrid** pattern recognition for ILD@FCC: TPC-seeded (`Clupatra`), build tracks from TPC hits, search for Si hits near tracks. New alg under R&D to combine Clupatra & conformal tracking (more details)



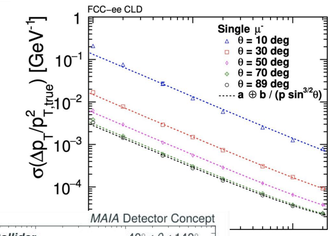
  - **With ML**: **Geometric Graph Track Finder**, embeds hits from subdetectors in latent space and clusters them. Available in key4hep (`GGTFTrackFinder`), model trained on IDEA

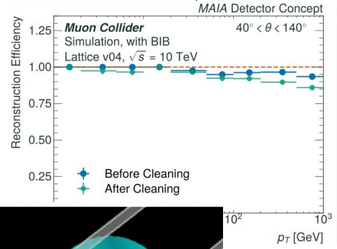


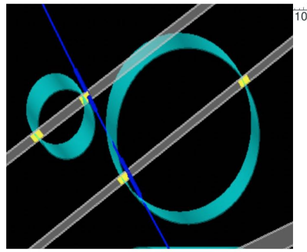
- More details in Andrea's talk at FCC week

Reconstruction - tracking 2/2

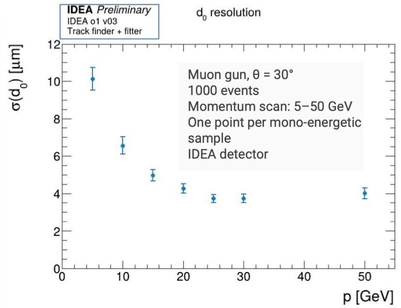
- **Track fitting**:
  - **Conformal tracking**: apply Kalman filter to tracks found from pattern recognition. Can be used for Si-based trackers. In key4hep (`ConformalTracking`), in use by **CLD** (more details)
  - **ACTS**: provides KF-based fitting, but also Gaussian Sum Filter for electron tracks (strongly non-Gaussian energy loss from bremsstrahlung), and Global-chi2 fitter for DCH tracks (work ongoing on Deterministic Annealing Fitter). In key4hep (`k4ActsTracking`), first validation with **MAIA** (more details)
  - GenFit2: framework developed for Panda/Belle-II, implements DAF, resolves left-right ambiguities. Available in key4hep (`GenfitTrackFitter`), included in ALLEGRO and IDEA reco chain (more details)





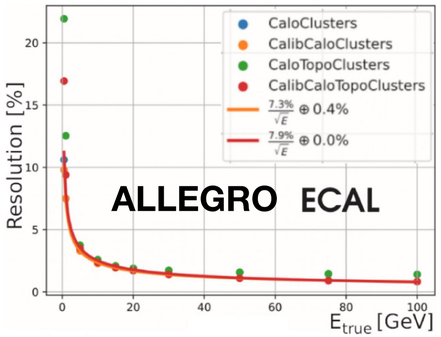


- **Vertexing**:
  - **Least-squares vertex fitter** code in **Delphes** (`VertexFit`) being migrated to Gaudi for **ALLEGRO/IDEA** (details)
  - Full-suite of algs in **LCFIPlus** developed for ILC and available via Marlin wrapper (used by **CLD** and **ILD**) (details)
    - **Tear-down PV finder, build-up SV finder, V0 fitting and selection**. Being ported to key4hep, reusing VertexFit for every trial fit (details), also for other detector concepts
  - Potential other candidates: **ACTS**
- **Benchmarking/validation**: algorithm developed and recently made available in **k4DetectorPerformance** (`TrackingValidationPlots`) (included as option in **ALLEGRO** reco chain)

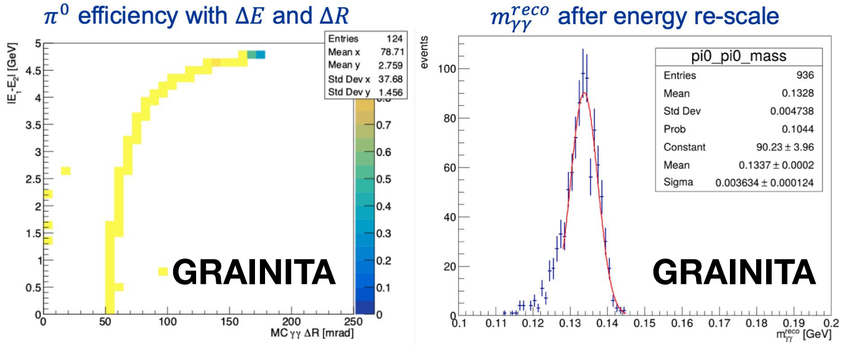


Reconstruction - clustering

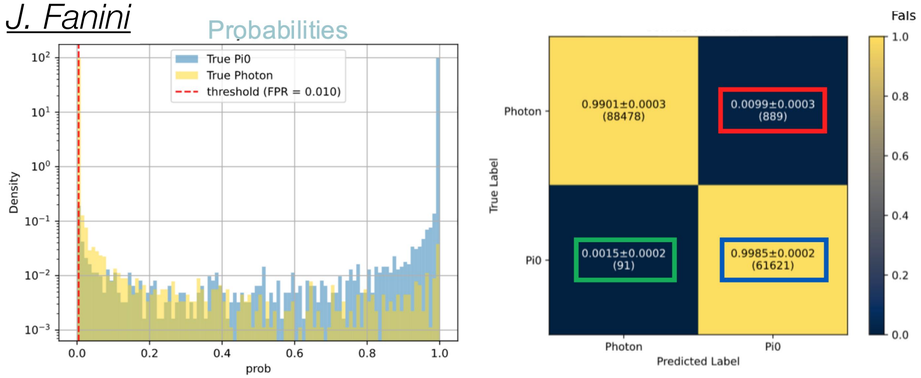
- Algorithms implemented for clustering of hits with sufficient deposited energy
  - **fixed-size clustering** (`CreateCaloClustersSlidingWindowFCCee`) finding local maxima of total energy in **sliding window**
  - **topological clustering** (`CaloTopoClusterFCCee`) of adjacent cells with S/B above given thresholds
  - **CLUE** (`k4Clue`): fast and parallel, density-based, weighted clustering algorithm - used e.g. for studies of the GRAINITA ECAL for **ALFA**



- Higher-level algorithms developed for e.g MVA-based **cluster energy regression, particle ID** (single photon vs pi0–>γγ) (BDT, transformer), ..

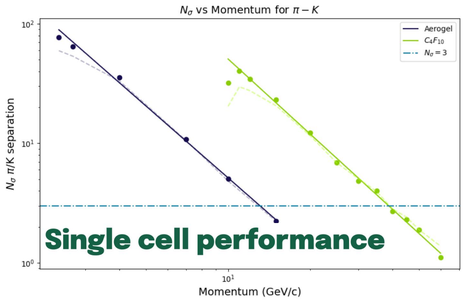


J. Fanini



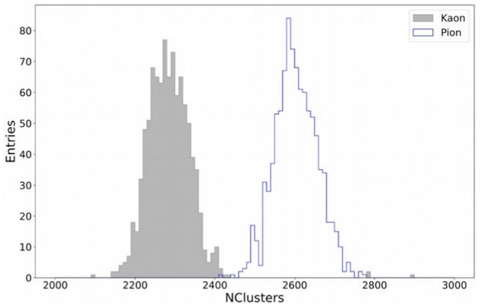
Reconstruction - higher-level

- **Global event reconstruction** (tracks+clusters => particle candidates with 4-momentum and PID)
  - **Particle-flow** (Pandora & ML-based) => see talk by Dolores tomorrow



- **PID** algorithms
  - **ARC**: algorithm (`ARCAlg`) for reconstruction of **Cherenkov** angle from track direction and hit positions under development (details). Simulations done at single-cell level, moving to full detector, to extract tabulated resolutions for π, K and p as a function of particle momentum and pseudorapidity for particle-flow studies
  - **dN/dx**: total track dN/dx can be computed from n(Clusters) of associated track hits and track length in **gaseous detectors**. More realistic dN/dx to be available after DCH digitiser v03 ready with digital pulse and peak-finding algorithm implemented for estimation of primary ionisation clusters

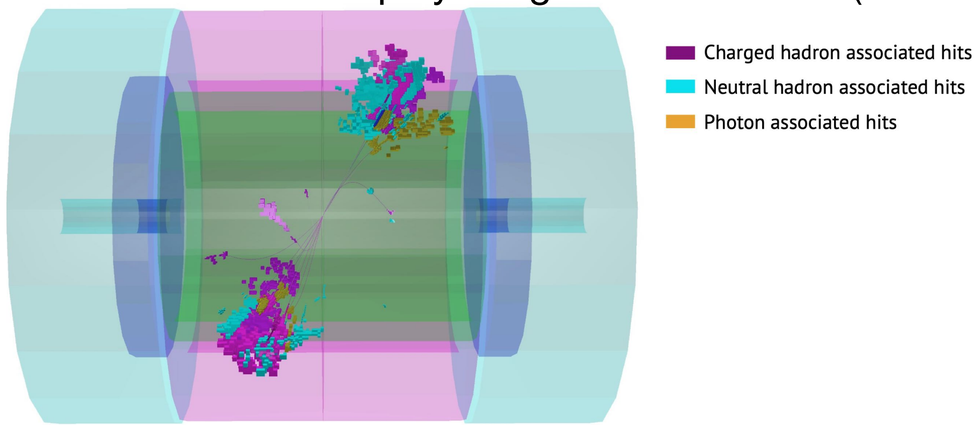
Clusters of pions and kaons at 7 GeV with perfect counting



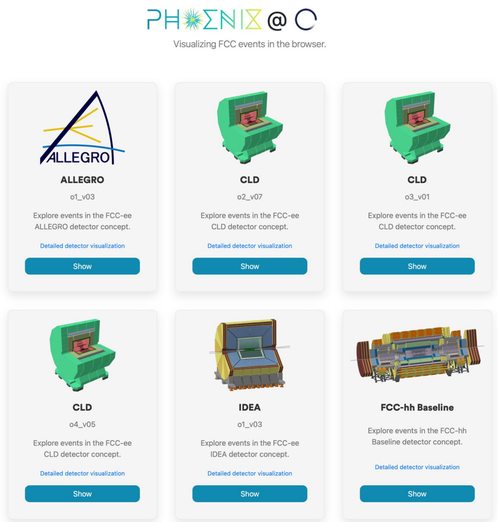
- **Jet** reconstruction and tagging
  - **clustering**: `ClusterJet` - basically a wrapper for FastJet
  - ML-based **flavour tagging**: transformer-based algorithm for **CLD** implemented in key4hep (`k4MLJetTagger`), expect more to come in the future with dedicated working group now established. See also tomorrow's talk by Gregor

Visualisation

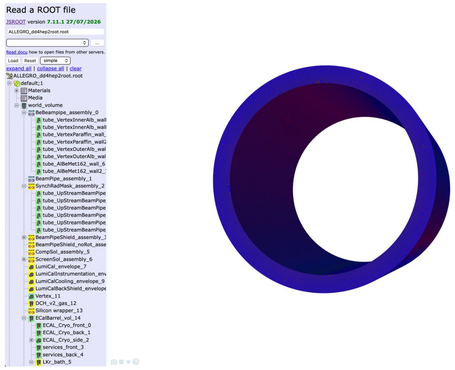
- A range of tools covering outreach, debugging, and geometry inspection
- Generic event display integrated in Pandora(Monitoring)



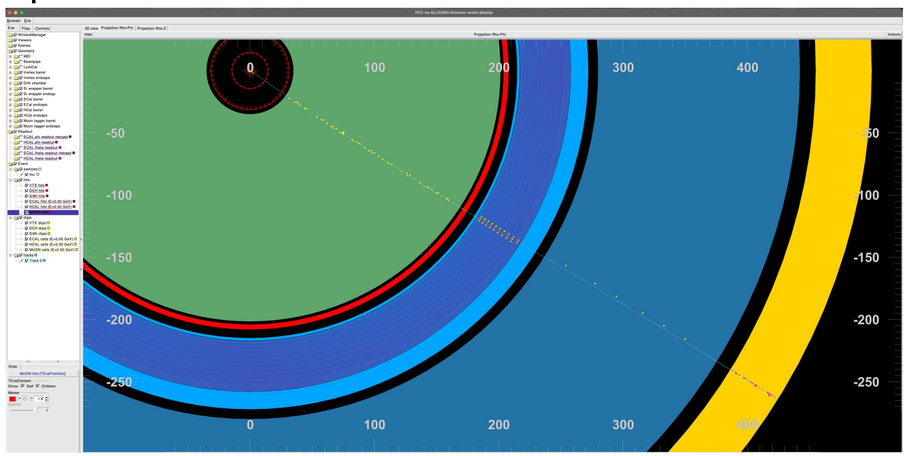
- Phoenix: web-based 3D event display mainly for outreach



- JSROOT: web-based, part of ROOT, primarily draws histograms and TTrees, but can also display/inspect detector geometries



- ALLEGRO-oriented event display (calodisplay) - for debugging, can visualise hits/digis/tracks/clusters/MC particles



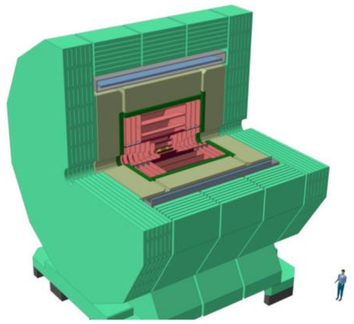
Meetings

- **FCC full-sim working meeting**
  - https://indico.cern.ch/category/16938/
  - Bi-weekly, on Wednesday, typically alternating between 11 am and 4 pm (CERN time)
  - Usually on lower-level aspects of the full simulation (geometry/digitisation/simple object reconstruction)

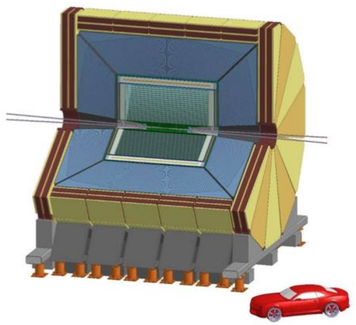
- **Reconstruction for FCC working meeting**
  - https://indico.cern.ch/category/19516/
  - Bi-weekly, on Thursday at 4 pm (CERN time)
  - On high-level event reconstruction

Conclusion

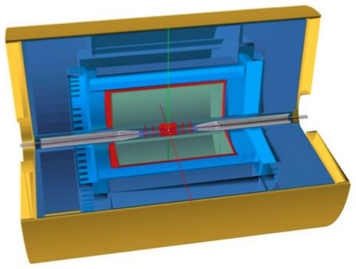
- The FCC **detector implementation** in the **full-simulation** is progressing **well**, with **increasingly sophisticated** digitisers and reconstruction algorithms being developed
- Existing algorithms may **benefit** from **further refinement** and could in some case require **adaptation** to the conditions of a circular collider.
- In parallel, several new **hardware developments** are underway within the DRDs and in the detector concepts
  - **effort** needed to **integrate** them into the central Key4HEP framework, but **facilitated** by **common** foundations and tools
- There is still **plenty** of **work ahead**, including validation, code optimisation, and detector optimisation
  - *additional **contributions** are always very **welcome**!*



CLD

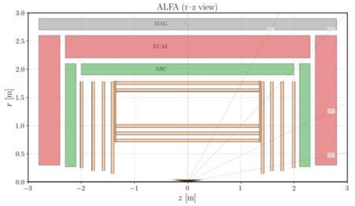


IDEA

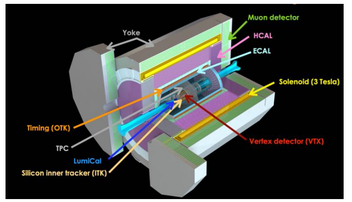


ALLEGRO

ILD



ALFA



AGORA