CLIC detector model and software tools for simulation and reconstruction

Muon Collider Workshop 2019

Marko Petric

On behalf of the CLICdp collaboration

CERN, 11 October 2019

CLIC – Compact Linear Collider

- High-luminosity linear e^+e^- collider
- The only mature multi-TeV option for future lepton colliders

Current scenario:

Stage 1: 1 ab^{-1} @ $t\bar{t}$/380 GeV

Precision SM Higgs and top physics

Stage 2: 2.5 ab^{-1} @1.5 TeV:

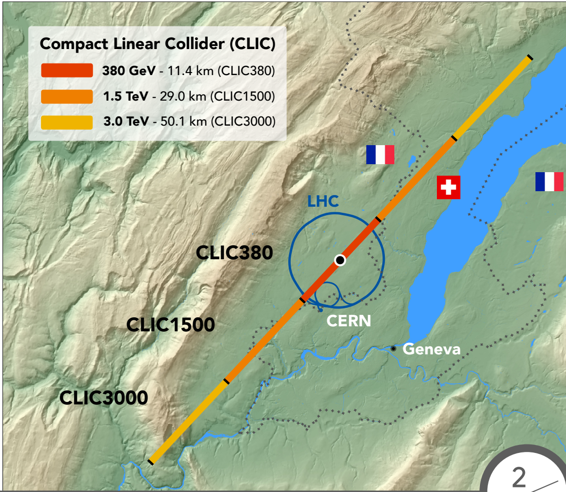
BSM, top and rare Higgs processes

Stage 3: 5 ab^{-1} @ 3 TeV :

BSM, top and rare Higgs processes

Each stage corresponds to 7-8 years

https://clic.cern



CLIC physics and tracking performance requirements

- Momentum resolution

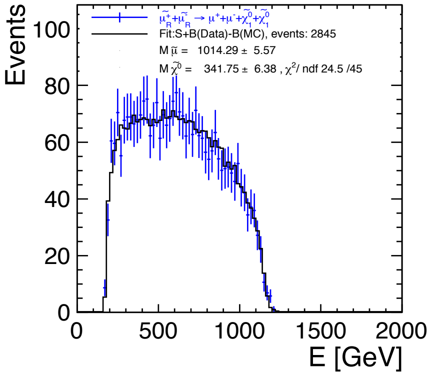
  Higgs recoil, $H \rightarrow \mu\mu$ or ℓ from BSM
  $\frac{\sigma(p_T)}{p_T^2} \sim 2 \times 10^{-5}\ \text{GeV}^{-1}$

- Jet energy resolution

  $W/Z/H$ di-jet separation
  $\frac{\sigma(E)}{E} \sim 3.5 - 5\%$
  for $E = 1000 - 50$ GeV

- Impact parameter resolution

  b/c tagging, Higgs couplings
  $\sigma_{r\phi} = \sqrt{a^2 + b^2 \cdot \text{GeV}^2/(p^2 \sin^3\theta)}$
  with $a = 5\ \mu$m and $b = 15\ \mu$m



CLIC physics and tracking performance requirements

- Momentum resolution

  Higgs recoil, $H \to \mu\mu$ or ℓ from BSM

  $\frac{\sigma(p_T)}{p_T^2} \sim 2 \times 10^{-5}\ \text{GeV}^{-1}$
- Jet energy resolution

  $W/Z/H$ di-jet separation

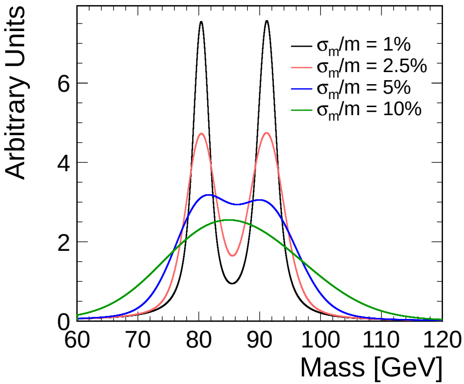
  $\frac{\sigma(E)}{E} \sim 3.5 - 5\%$

  for $E = 1000 - 50$ GeV
- Impact parameter resolution

  b/c tagging, Higgs couplings

  $\sigma_{r\phi} = \sqrt{a^2 + b^2 \cdot \text{GeV}^2/(p^2 \sin^3\theta)}$

  with $a = 5\ \mu$m and $b = 15\ \mu$m



CLIC physics and tracking performance requirements

- Momentum resolution

  Higgs recoil, $H \to \mu\mu$ or ℓ from BSM

  $\frac{\sigma(p_T)}{p_T^2} \sim 2 \times 10^{-5}\ \mathrm{GeV}^{-1}$

- Jet energy resolution

  $W/Z/H$ di-jet separation

  $\frac{\sigma(E)}{E} \sim 3.5 - 5\%$

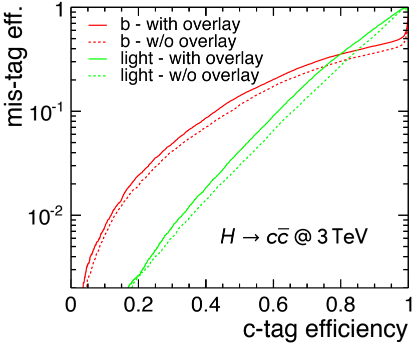
  for $E = 1000 - 50$ GeV

- Impact parameter resolution

  b/c tagging, Higgs couplings

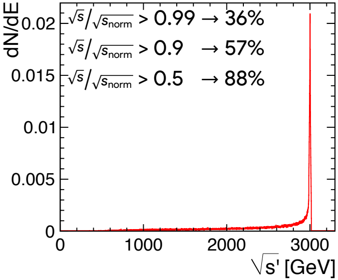
  $\sigma_{r\phi} = \sqrt{a^2 + b^2 \cdot \mathrm{GeV}^2/(p^2 \sin^3\theta)}$

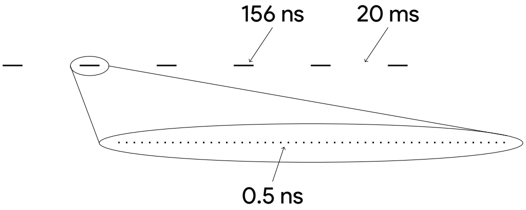
  with $a = 5\ \mu$m and $b = 15\ \mu$m



CLIC beam structure and beam-induced backgrounds

- Time stamp hits from the detectors (central det ⟺ physics event)
- Tracking timing requirements
  - 10 ns integration time
  - $10/\sqrt{12}$ ns hit resolution
- Background overlay (10 (20) beam crossings before (after) physics event)

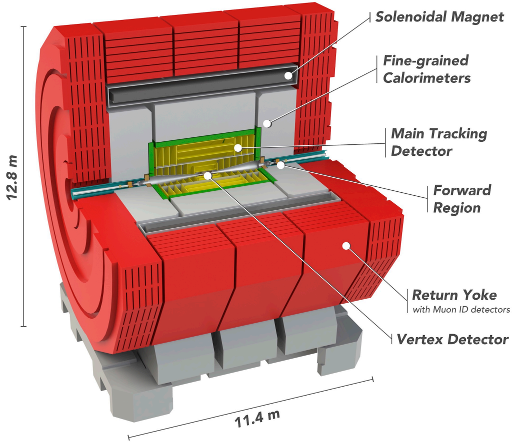




CLIC: trains at 50 Hz, 1 train = 312 bunches

- entire bunch train available for offline reconstruction
- not all bunches crossings contain a “hard” interaction

CLIC Detector Concept

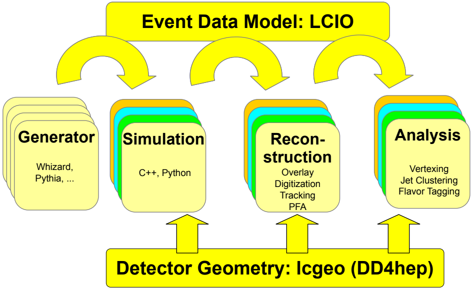


- B Field of 4 T
- Vertex: 3 double layers
  - Single-hit accuracy 3 µm
  - $0.2\% X_0$ per detection layer
  - Power pulsing → 50 mW cm^{-2}
- Si tracker: $r_{max} = 1.5$ m
  - single point resolution = 7 µm × 90 µm
  - detector: ~ $1\% X_0$ per layer
  - support & cables: ~ $2.5\% X_0$
- Precise timing for background rejection
  - 10 ns stamping for tracks
  - 1 ns accuracy for calo. cluster

Low-mass → small multiple scattering

Linear Collider Software

- Linear collider community has used and developed common software for many years
  - Event data model (EDM) and persistency: LCIO
    - PODIO is being investigated in AIDA++
  - Particle flow reconstruction: PandoraPFA
- Adopted DD4hep geometry description to develop more common software this geometry information
- Interface generic reconstruction packages via thin wrappers to linear collider framework

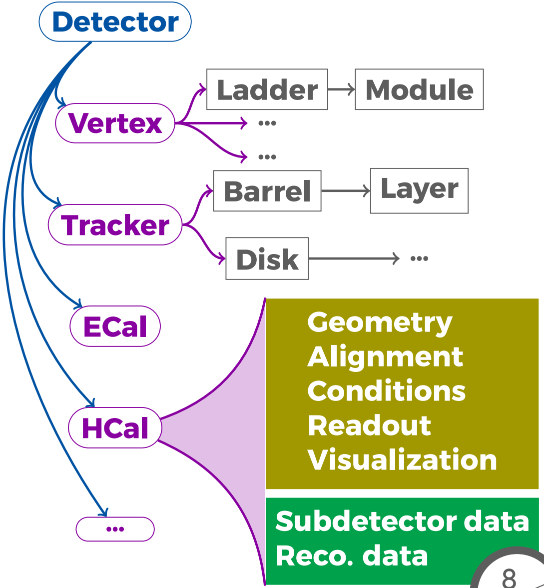


DD4hep

- Complete Detector Description
  - Providing geometry, materials, visualization, readout, alignment, calibration
- Supports full experiment life cycle
  - Detector concept development, detector optimization, construction, operation
  - Facile transition from one stage to the next
- Single source of information → consistent description
  - Use in simulation, reconstruction, analysis
- Ease of Use
- Few places for entering information
- Minimal dependencies
- AIDA-2020 and HSF member project

What is Detector description?

- Description of a tree-like hierarchy of 'detector elements'
  - Sub-detectors or parts of subdetectors
- Detector Element describes:
  - Geometry
    - points to placed logical volumes
  - Environmental conditions
  - Properties required to process event data
  - Extenstions (optionally): experiment, sub-detector or activity specific data, measurement surfaces



DD4hep – Generic Detector Palette

- Generic driver available → scalable and flexible
- Parameters are provided in compact XML files, e.g.

```
<detector id="15" name="HCal" type="GenericCalBarrel_o1_v01" readout="HCalCollection">
  <envelope vis="HCALVis">
    <shape type="PolyhedraRegular" numsides="HCal_sym" rmin="HCal_rmin" rmax="HCal_rmax" dz="HCal_dz" material="Air"/>
    <rotation x="0*deg" y="0*deg" z="90*deg-180*deg/HCal_symmetry"/>
  </envelope>
  <dimensions numsides="HCal_sym" rmin="HCal_rmin" z="HCal_dz*2"/>
  <layer repeat="(int) HCal_layers" vis="HCalLayerVis">
    <slice material="Steel235"    thickness="0.5*mm" vis="HCalAbsorberVis" radiator="yes"/>
    <slice material="Steel235"    thickness="19*mm"  vis="HCalAbsorberVis" radiator="yes"/>
    <slice material="Polystyrene" thickness="3*mm"   sensitive="yes"  limits="cal_limits"/>
    <slice material="Copper"      thickness="0.1*mm" vis="HCalCopperVis"/>
    <slice material="PCB"         thickness="0.7*mm" vis="HCalPCBVis"/>
    <slice material="Steel235"    thickness="0.5*mm" vis="HCalAbsorberVis" radiator="yes"/>
    <slice material="Air"         thickness="2.7*mm" vis="InvisibleNoDaughters"/>
  </layer>
</detector>
```

- You can scale, change layers, radii and compositions...
- Propagate visualization attributes to Display
- Inspect `${DD4hep_ROOT}/DDDetectors/src` or `compact`

Your Detector Palette

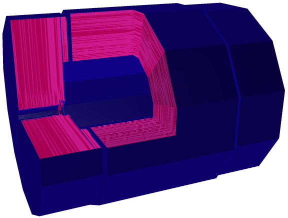
```
static Ref_t create_detector(Detector& theDetector,
                             xml_h e, SensitiveDetector sens) {

  xml_det_t x_det = e;
  Layering layering(x_det);
  xml_comp_t staves = x_det.staves();
  xml_dim_t dim = x_det.dimensions();
  DetElement sdet(det_name, x_det.id());
  Volume motherVol = theDetector.pickMotherVolume(sdet);

  PolyhedraRegular polyhedra(numSides, rmin, rmax, detZ);
  Volume envelopeVol(det_name, polyhedra, air);

  for (xml_coll_t c(x_det, _U(layer)); c; ++c) {
    xml_comp_t x_layer = c;
    int n_repeat = x_layer.repeat();
    const Layer* lay = layering.layer(layer_num - 1);
    for (int j = 0; j < n_repeat; j++) {
      string layer_name = _toString(layer_num, "layer%d");
      double layer_thickness = lay->thickness();
      DetElement layer(stave, layer_name, layer_num);
        ...}
  }
  DECLARE_DETELEMENT(GenericCalBarrel_o1_v01, create_detector)
```

- Users can easily write their own detector drivers, if needed
- Detector geometry extendable with additional info.
- C++ model of separation of ‘data’ and ‘behavior’
  - Classes consist of a single ‘reference’ to the data object



DDG4 – Gateway to Geant4

- In-memory translation of geometry from TGeo to Geant4
  - Materials, Solids, Limit sets, Regions
  - Logical volumes, placed volumes and physical volumes
- External configuration:
  - Plugin mechanism
  - Property mechanism to configure plugin instances
  - Supports configuration via XML, Python or ROOT-AClick
- Use plugin mechanism to configure: Generation, Event Action, Tracking Action, Stepping Action, SensitiveDetector, PhysicsList...
- Provides out of the box MC truth handling with record reduction

DDG4 – Configuration example

- DDG4 is highly modular
- Very easily configurable through python

```
#...
gen = DDG4.GeneratorAction( kernel , "LCIOInputAction/LCIO1" )
gen.Input = "LCIOFileReader|" + inputFile
#...
```

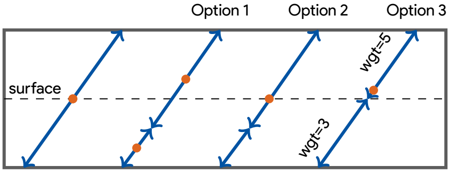
- Or configure actions, filters, sequences, cuts

```
#...
part = DDG4.GeneratorAction(kernel, "Geant4ParticleHandler/ParticleHandler")
kernel.generatorAction().adopt(part)
part.SaveProcesses = ['Decay']
part.MinimalKineticEnergy = 1*MeV
part.KeepAllParticles = False
#...
user = DDG4.GeneratorAction(kernel,"Geant4TCUserParticleHandler/UserParticleHandler")
user.TrackingVolume_Zmax = DDG4.tracker_region_zmax
user.TrackingVolume_Rmax = DDG4.tracker_region_rmax
#...
```

DD4hep – Plugin Palettes

- Providing input handlers, sensitive detectors for most cases…
- Hard to provide Geant4 Sensitive Detectors for all cases
  - Couples detector 'construction' to reconstruction, MC truth and Hit production
  - Too dependent on technology and user needs

e.g. several possibilities for tracker



  - Providing palette of most 'common' sensitive components for trackers and calorimeters
- Physics lists, Physics/particle constructors etc.
  - Wrapped factory plugins directly taken from Geant4
  - Users extend physics list (e.g. QGSP)
- Several IO handlers (LCIO, ROOT, StdHep, HepEvt, HepMC)

A bit more about ddsim

- Get steering file `ddsim --dumpSteeringFile > mySteer.py`
  - Steering file includes documentation for parameters and examples
  - The python file contains a `DD4hepSimulation` object at global scope
  - Configure simulation directly from commandline

```
from DDSim.DD4hepSimulation import DD4hepSimulation
from SystemOfUnits import mm, GeV, MeV, keV
SIM = DD4hepSimulation()
SIM.compactFile = "CLIC_o3_v06.xml"
SIM.runType = "batch"
SIM.numberOfEvents = 2
SIM.inputFile = "electrons.HEPEvt"
SIM.part.minimalKineticEnergy = 1*MeV
SIM.filter.filters ['edep3kev'] =
dict (name="EnergyDepositMinimumCut/3keV" ,
      parameter={"Cut" : 3.0*keV} )
```

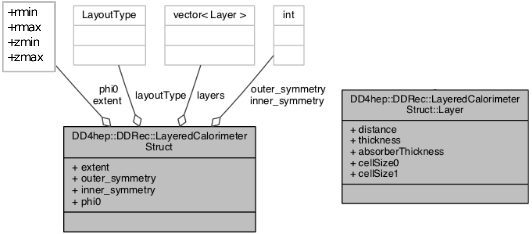
```
$ ddsim
--action.calo                    --filter.tracker          --part.keepAllParticles
--action.mapActions              -G                        --part.minimalKineticEnergy
--action.tracker                 --gun.direction           --part.printEndTracking
--compactFile                    --gun.energy              --part.printStartTracking
--crossingAngleBoost             --gun.isotrop             --part.saveProcesses
--dump                           --gun.multiplicity        --physics.decays
--dumpParameter                  --gun.particle            --physics.list
--dumpSteeringFile               --gun.position            --physicsList
--enableDetailedShowerMode       -h                        --physics.rangecut
--enableGun                      --help                    --printLevel
--field.delta_chord              -I                        --random.file
--field.delta_intersection       --inputFiles              --random.luxury
--field.delta_one_step           -M                        --random.replace_gRandom
--field.eps_max                  --macroFile               --random.seed
--field.eps_min                  -N                        --random.type
--field.equation                 --numberOfEvents          --runType
--field.largest_step             -O                        -S
--field.min_chord_step           --outputFile              --skipNEvents
--field.stepper                  --output.inputStage       --steeringFile
--filter.calo                    --output.kernel           -v
--filter.filters                 --output.part             --vertexOffset
--filter.mapDetFilter            --output.random           --vertexSigma
```

DDRec: High Level Information

High level view onto the detectors through DDRec `DataStructures` extensions for `DetElements`

- Constructors fill DDRec `DataStructures`
- `DataStructures` allow to decouple detector implementation from reconstruction algorithms

`DataStructures` contain sufficient information to provide geometry information to particle flow clustering via `PandoraPFA`

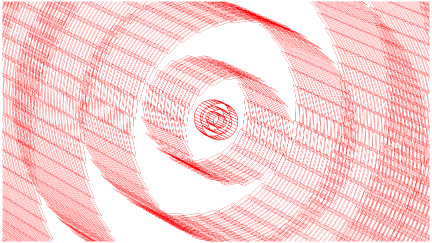


Data Structure	Detector Type
`ConicalSupportData`	Cones and Tubes
`FixedPadSizeTPCData`	Cylindrical TPC
`LayeredCalorimeterData`	Sandwich Calorimeters
`ZPlanarData`	Planar Silicon Trackers
`ZDiskPetalsData`	Forward Silicon Trackers

Geometry for Track Reconstruction

Information needed for track reco.

- measurement of directions of hits
- local-to-global transforms
- material properties

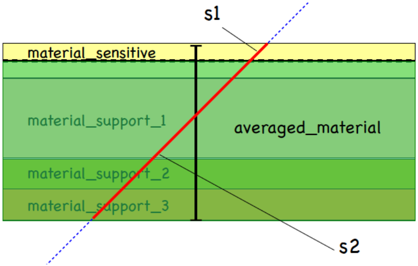


DD4hep surfaces provide this

- Surfaces can be auto. added

```
<plugin name="DD4hep_GenericSurfaceInstallerPlugin">
  <argument value="TrackDet"/>
  <argument value="dimension=2"/>
  <argument value="u_x=-1."/>
  <argument value="v_y=-1."/>
  <argument value="n_z=1."/>
</plugin>
```

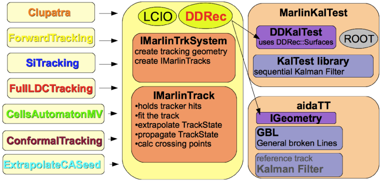
  - Plugin loops over all `DetElements`
  - Configure surface type and direction in volume
  - Automatically average materials

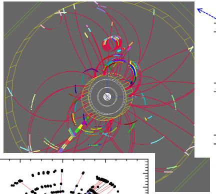


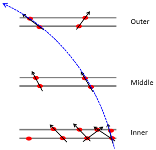
Track Reconstruction

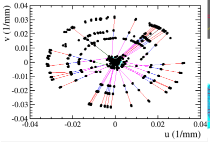
Track reconstruction using DD4hep surfaces

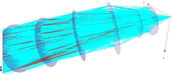
- Pattern recognition/track finding algorithms
  - From detector specific: *Clupatra* for TPC; mini-vector for vertex detector double layers
  - to geometry agnostic: pattern recognition in conformal space
- Track fitting, fairly generic: DDKalTest, aidaTT
  - ACTS might be long term replacement (AIDA++)
- Geometry: Interfaced via DDRec and Surfaces











DDKalTest using DD4hep surf. for track fitting

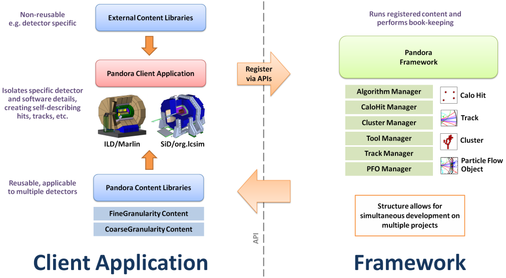
Particle Flow Reconstruction

PandoraPFA: generic toolkit for pattern recognition algorithms in highly granular calorimeters

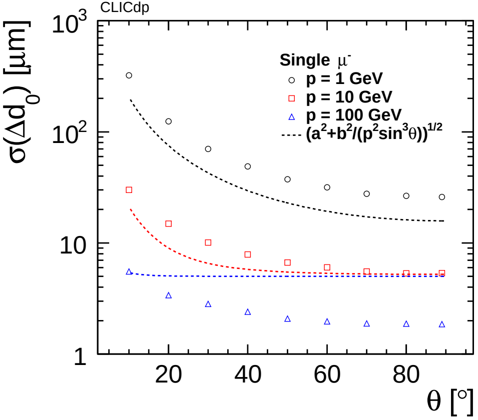
- Originally developed for ILC/CLIC
- Extended to work in LAr-TPC reconstruction for the DUNE experiment

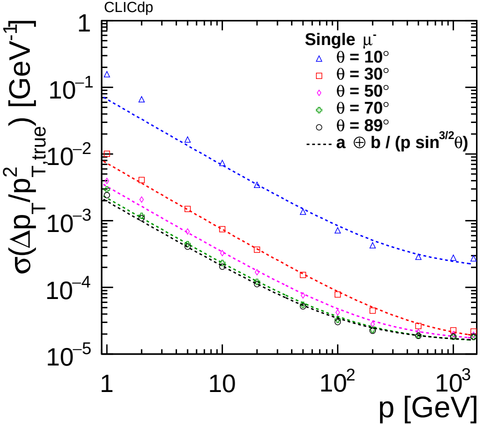
ClientApplication: DDMarlinPandora glues linear collider framework (Marlin), DD4hep, and PandoraPFA

- Passes DDRec `DataStructures` information, tracks, and calorimeter hits to PandoraPFA
- Converts PandoraPFA objects into LC EDM



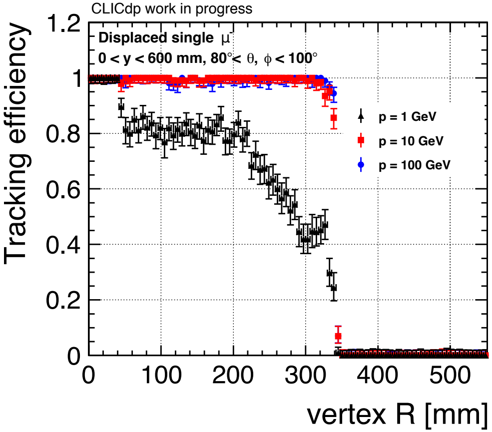
Performance d_0 and p_T resolution for single μ

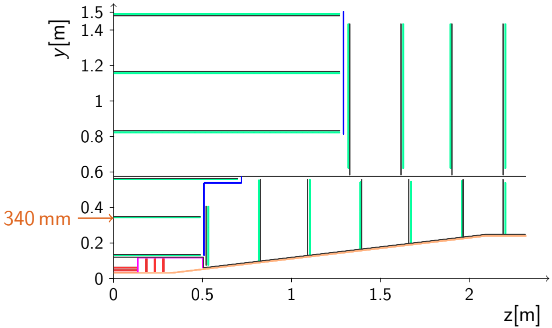




- In good agreement with target values of required performance

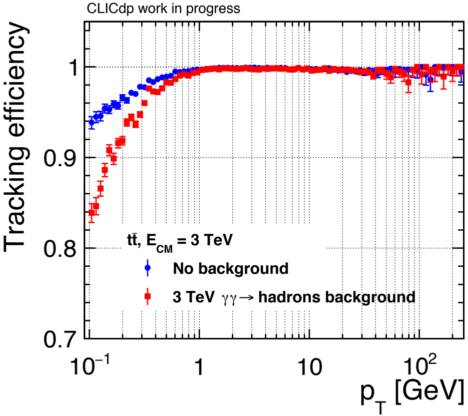
Efficiency for displaced single μ

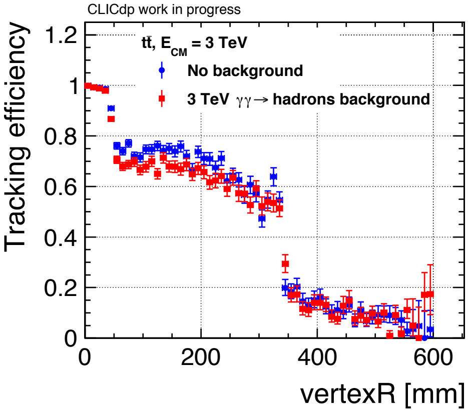




- Track generated uniformly along y axis with given opening angle

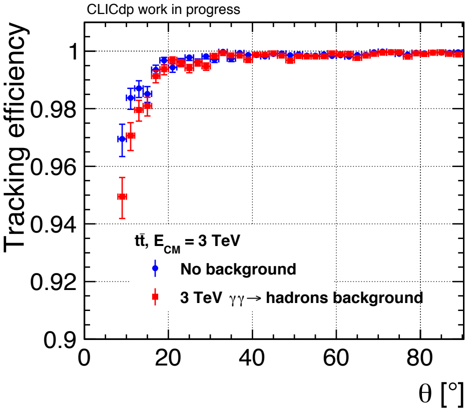
Efficiency for *tt*-bar @ 3 TeV

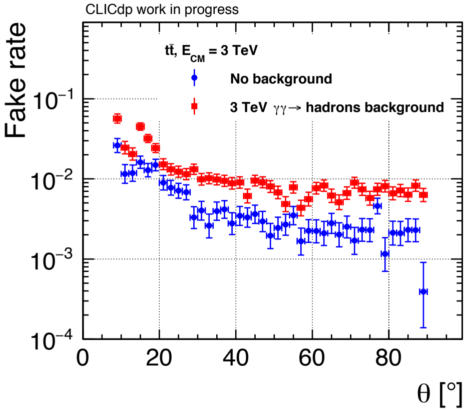




- Efficiency = pure reconstructed particles / reconstructable
  - pure = purity > 75% (hits belong to associated MC particles / total hits)
  - reconstructable = stable, p_T > 0.1 GeV, unique hits ≥ 4

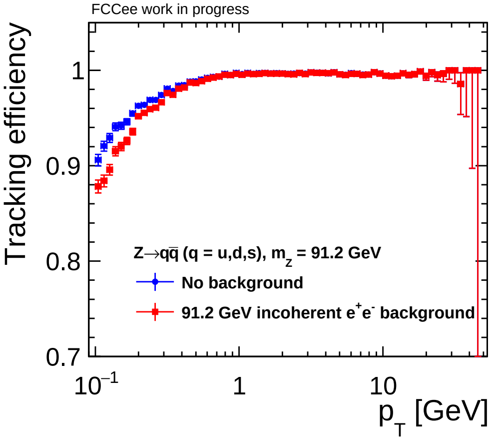
Efficiency and fake rate for *tt*-bar @ 3 TeV





- More performance plots https://cds.cern.ch/record/2649437

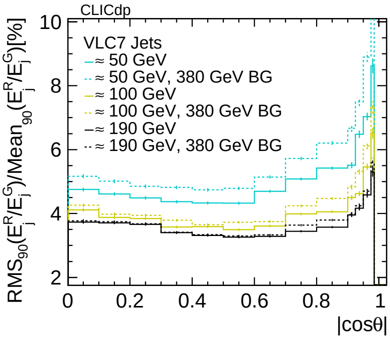
Works also with different geometry - FCCee

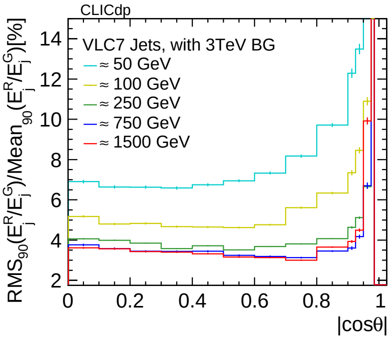


- Z bosons at rest decaying into light quarks
- background from e^+e^- pairs

- Works also with different detectors e.g. FCCee CLD
- using DD4hep Detector description
- Smaller magnetic field 4T→ 2T
- Larger tracker 1.5m→ 2.15m
- smaller beam-pipe 29mm→ 15mm
- adapting pattern recognition parameters

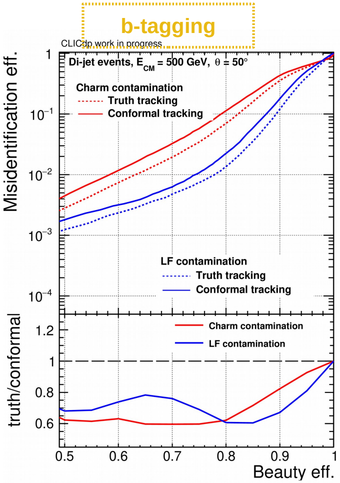
Jet Energy Resolution

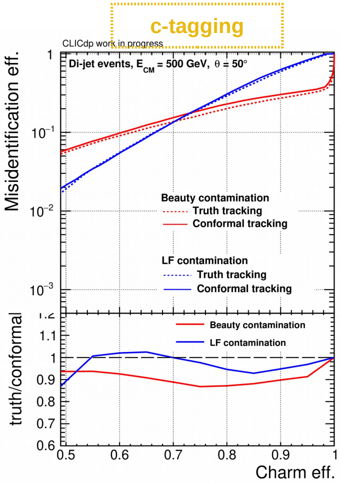




- Performance from light flavor di-jets
- Using PandoraPFA and FastJet (VLC R=0.7)
- In good agreement with target values of required performance
- More performance plots https://cds.cern.ch/record/2649437

Flavour Tagging





- Vertex finder reconstructs primary and secondary vertices
- Jet reconstruction using jet clustering algorithm
- Feed information into MVA to establish the flavourness of each jet

Summary and Conclusions

- CLIC offers a wealth of accurate e^+e^- physics measurements
- CLICdet detector design fully optimized on full simulation
- Validated software for detector optimization and physics studies from Geant4 simulation to user analysis
  - Based on DD4hep
  - Reconstruction aimed at Si tracking, Particle flow
- Plans at CERN to create more synergy between software used for CLIC and
  - Replace underlying framework
  - Provide wrappers for relevant processors
  - Core components: DD4hep; later maybe PODIO and ACTS
  - Keep our software chain running to continue full simulation studies
- Docs: https://clic.cern/european-strategy