Delphes 3
a modular framework for fast simulation of a generic collider experiment

Pavel Demin, Michele Selvaggi

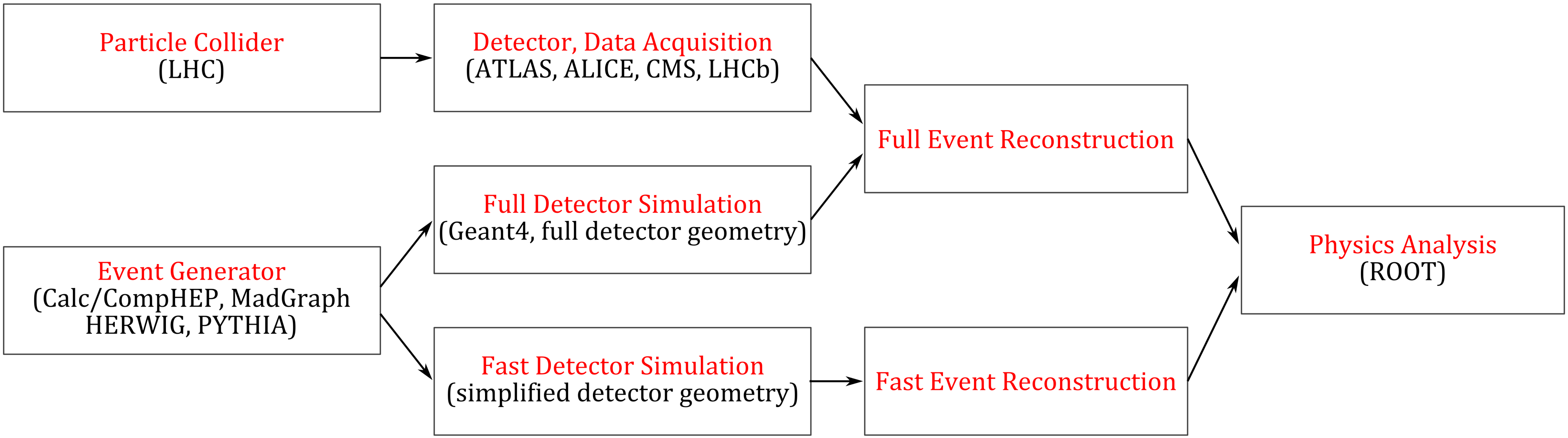
Université catholique de Louvain
Louvain-la-Neuve, Belgium

Cosmology, Particle Physics and Phenomenology (CP3)

July 24, 2014

What is Fast Simulation?

Detector Simulation



- **Full Detector Simulation** and **Full Event Reconstruction** is very detailed but slow

  ~ 10 – 1000 s/event
- **Fast Detector Simulation** and **Fast Event Reconstruction** is faster but less detailed
  - **Simplify** the slowest parts of the simulation (e.g., calorimeter response) and of the reconstruction (e.g., track reconstruction). Examples: Atlfast-II, CMS FastSim

    ~ 1 – 100 s/event
  - Or **parametrize** the whole response of the detector and of the reconstruction algorithms. Examples: Delphes, PGS

    ~ 0.01 – 1 s/event

What is Delphes?

Here's what the Wikipedia has to say about Delphes:

http://fr.wikipedia.org/wiki/Delphes

… Delphes (en grec: Δελφοί) est le site d'un sanctuaire panhellénique où parlait l'oracle d'Apollon à travers sa prophétesse, la Pythie…

Origin of Delphes

- Delphes project started back in 2007 at UCL as a side project to allow quick feasibility studies

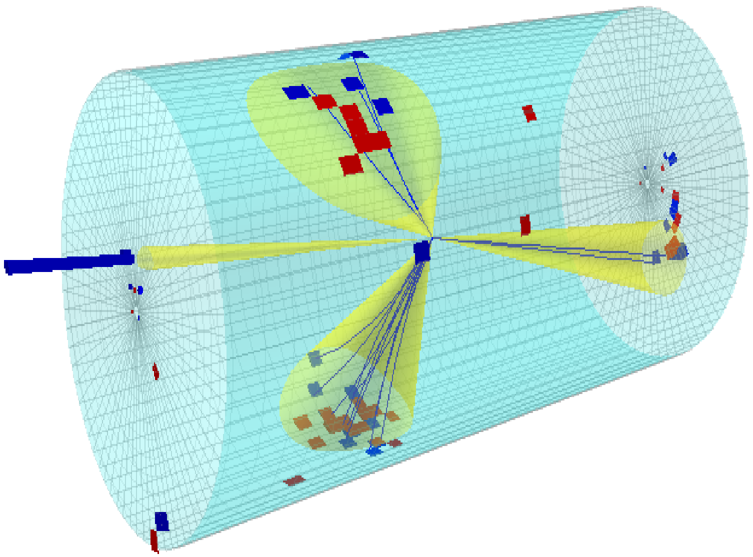
- Following the guidelines and suggestions of the 2012 LPCC workshop on public fast simulators for the LHC, Delphes has been completely redesigned to meet the needs of all users

- In 2013, Delphes 3 was released:
  - modular structure allowing users to easily introduce new features and modify existing ones
  - library interface to use Delphes inside other programs
  - simulation speed has been improved
  - input file readers have been rewritten from scratch
  - many existing features have been updated
  - important number of bug fixes

- Widely tested and used by the community (pheno, Snowmass, CMS ECFA efforts, etc)

- **Delphes** is a **modular framework** that **parametrizes** the response of a multipurpose detector and of the reconstruction algorithms

- The simulation includes
  - tracking system, embedded into a magnetic field,
  - calorimeters with electromagnetic and hadronic sections,
  - muon system,
  - very forward detectors arranged along the beam-line [JINST 2 (2007) P09005].



- It performs
  - propagation of stable particles,
  - "interaction" with the detector (parametric approach to efficiency and resolution convolution),
  - "reconstruction" of physics objects.

Useful Links

- Website:

  https://cp3.irmp.ucl.ac.be/projects/delphes

- Documentation:

  https://cp3.irmp.ucl.ac.be/projects/delphes/wiki/WorkBook

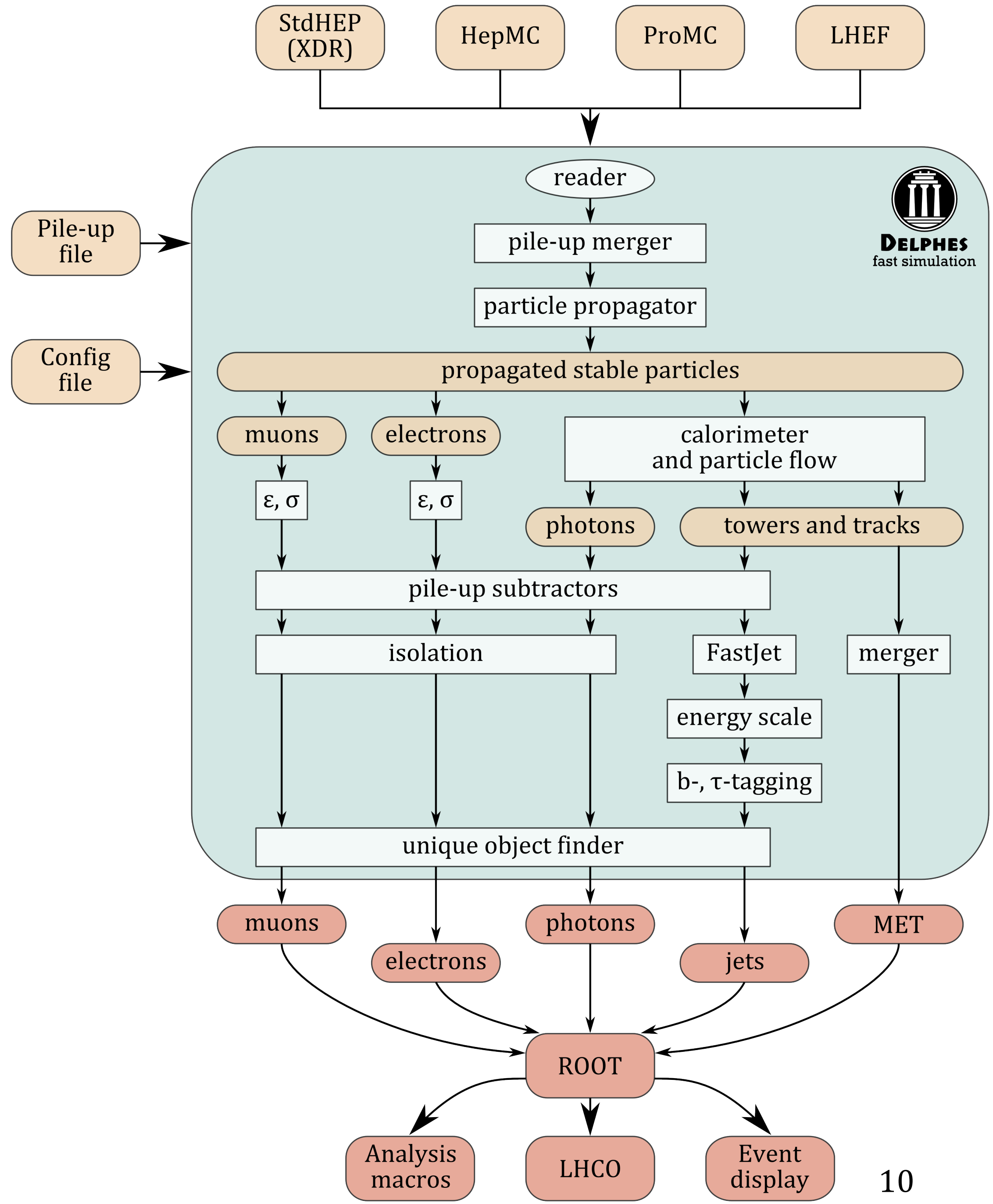
- Paper:

  http://dx.doi.org/10.1007/JHEP02(2014)057
  http://arxiv.org/abs/1307.6346

How Delphes Works?

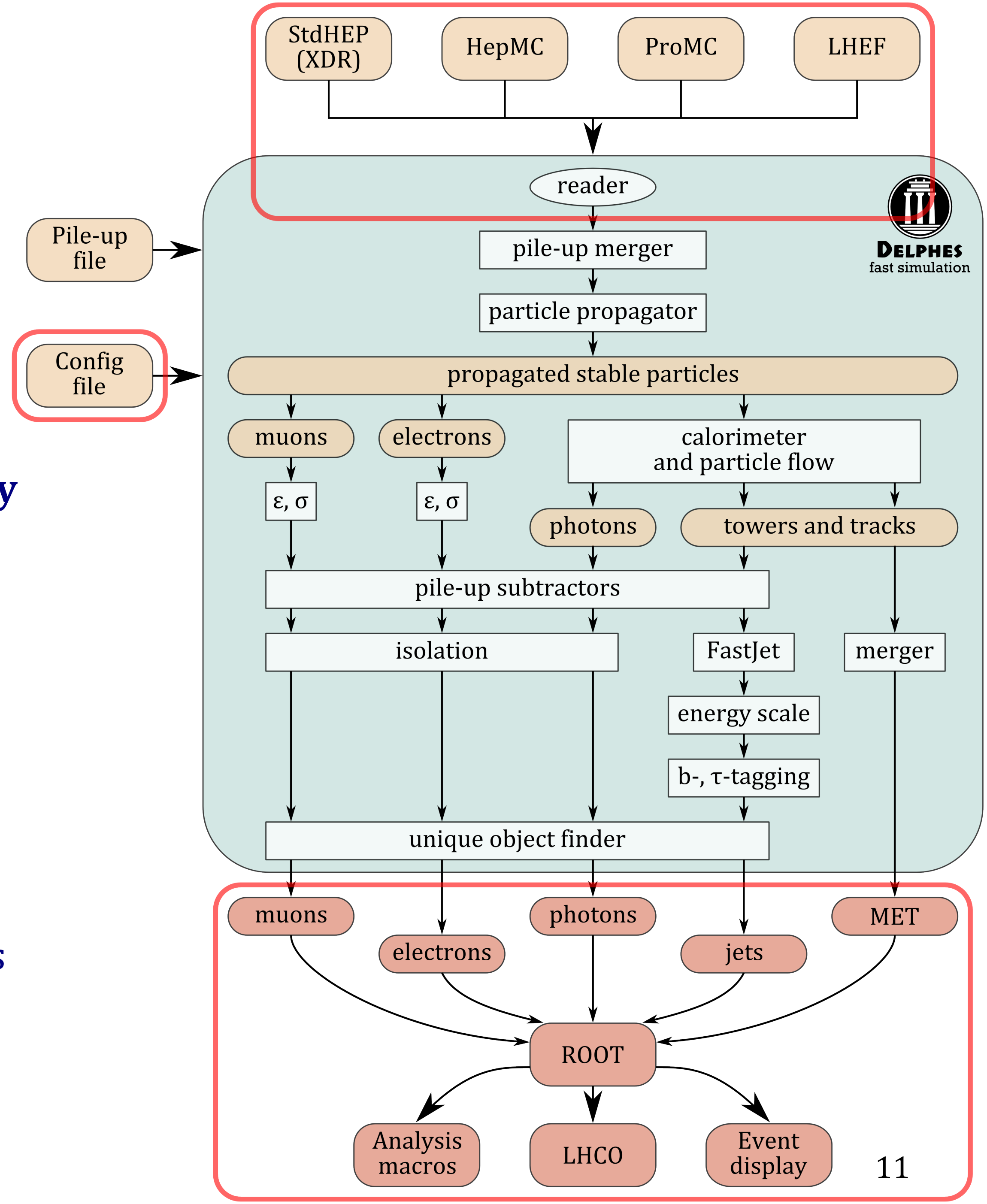
Modular Structure

- every physics object in Delphes is a **Candidate** (four-vector like object)
- all **modules** consume and produce **arrays** of **Candidates**
- modular system allows you to:
  - define your own output collections
  - store variants of object collections
  - define the isolation criteria for each type of object
  - define efficiency and resolution formulae for all objects
  - ...
- Delphes includes a set of modules and example configuration files well tested against expected response of ATLAS and CMS



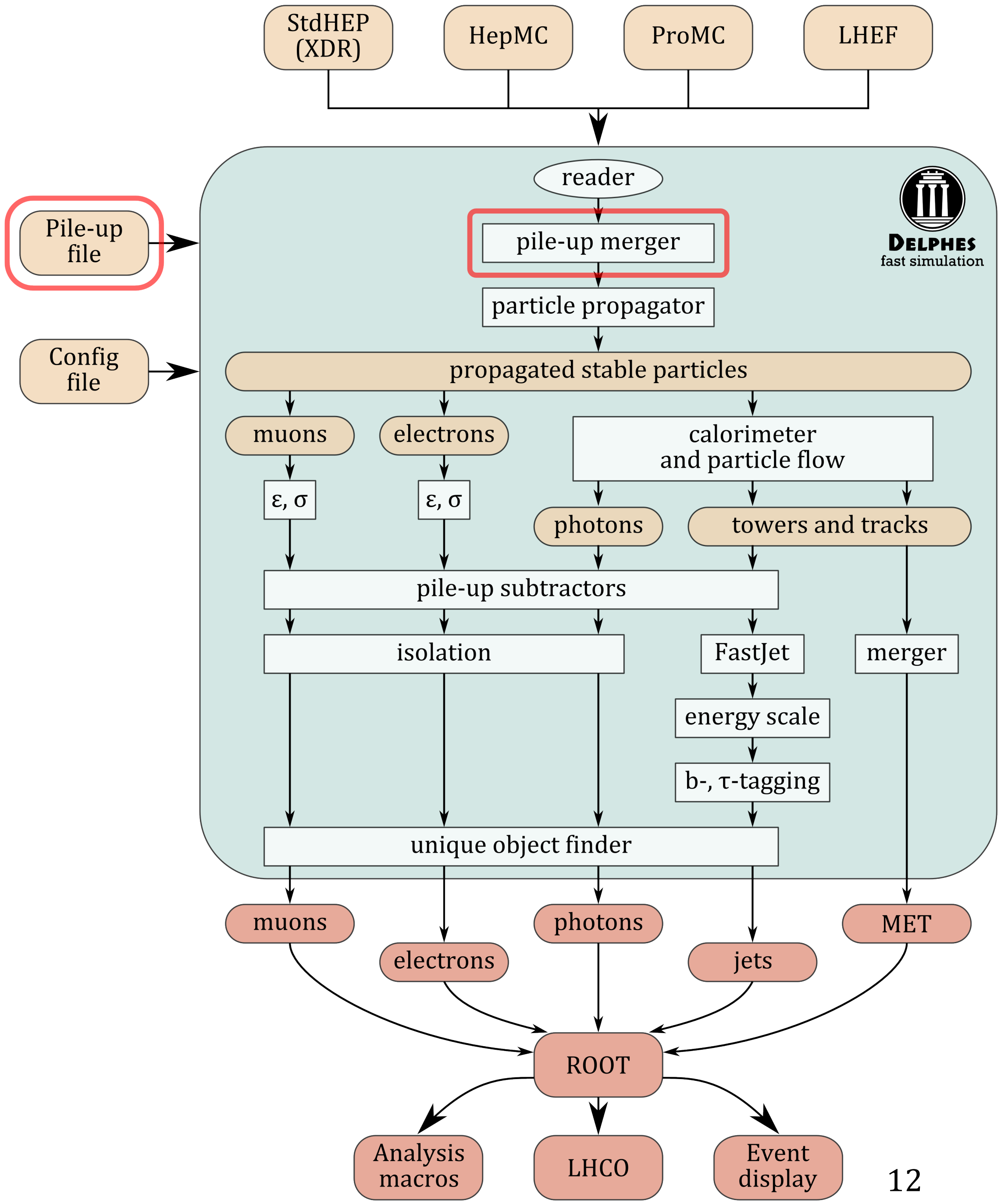
Input and Output

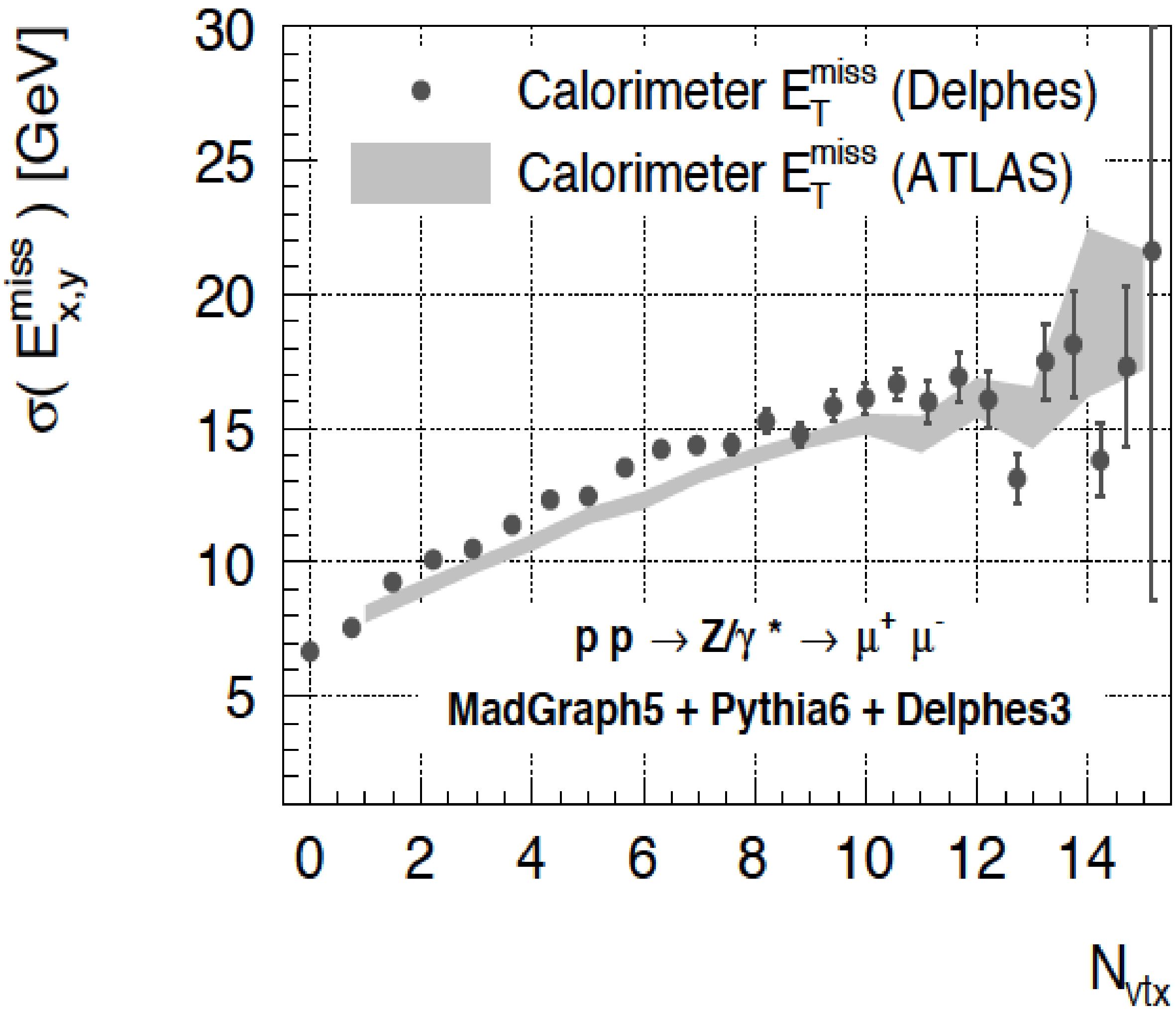
- Delphes' **input** is a list of particles produced by an event generator
- files in various formats can be read
- readers for new formats can be easily added
- Delphes also can be used as a **library** inside other programs
- Delphes' **output** is a ROOT tree containing the analysis objects
- **configuration file**:
  - modules interconnection, execution order and parameters
  - output object collections



Pile-up Mixing

- pile-up mixing is implemented in the following way:
  - a random (Poisonian) number of **pre-generated** minimum bias events is added to the main event
  - minimum bias events are
    - spread along z-axis
    - rotated by a random angle φ w.r.t. z-axis



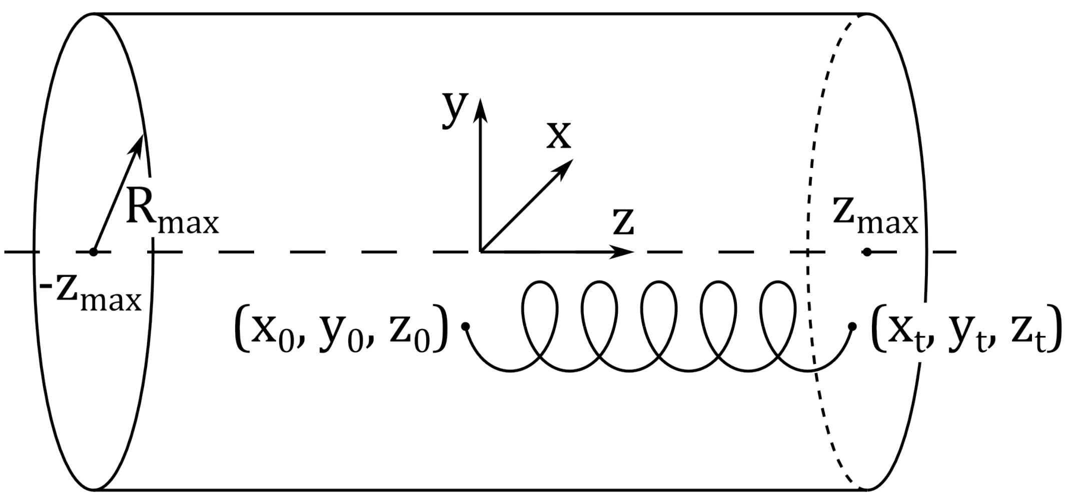


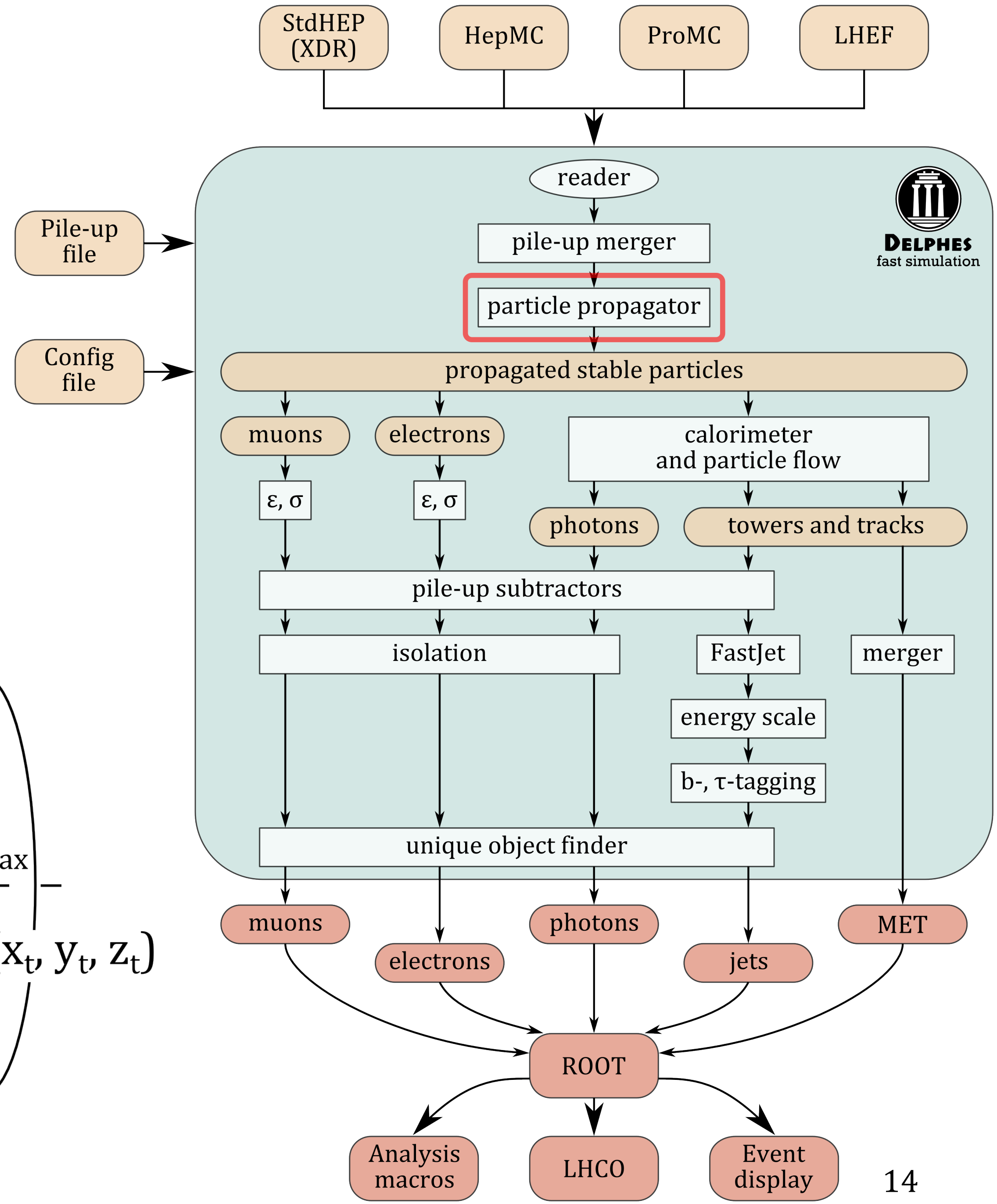
→ **good agreement**

Particle Propagation

- **charged** and **neutral** particles are propagated in a solenoidal magnetic field until they reach the calorimeters

- propagation parameters:
  - magnetic field (B)
  - radius and half-length (R_{max}, z_{max})



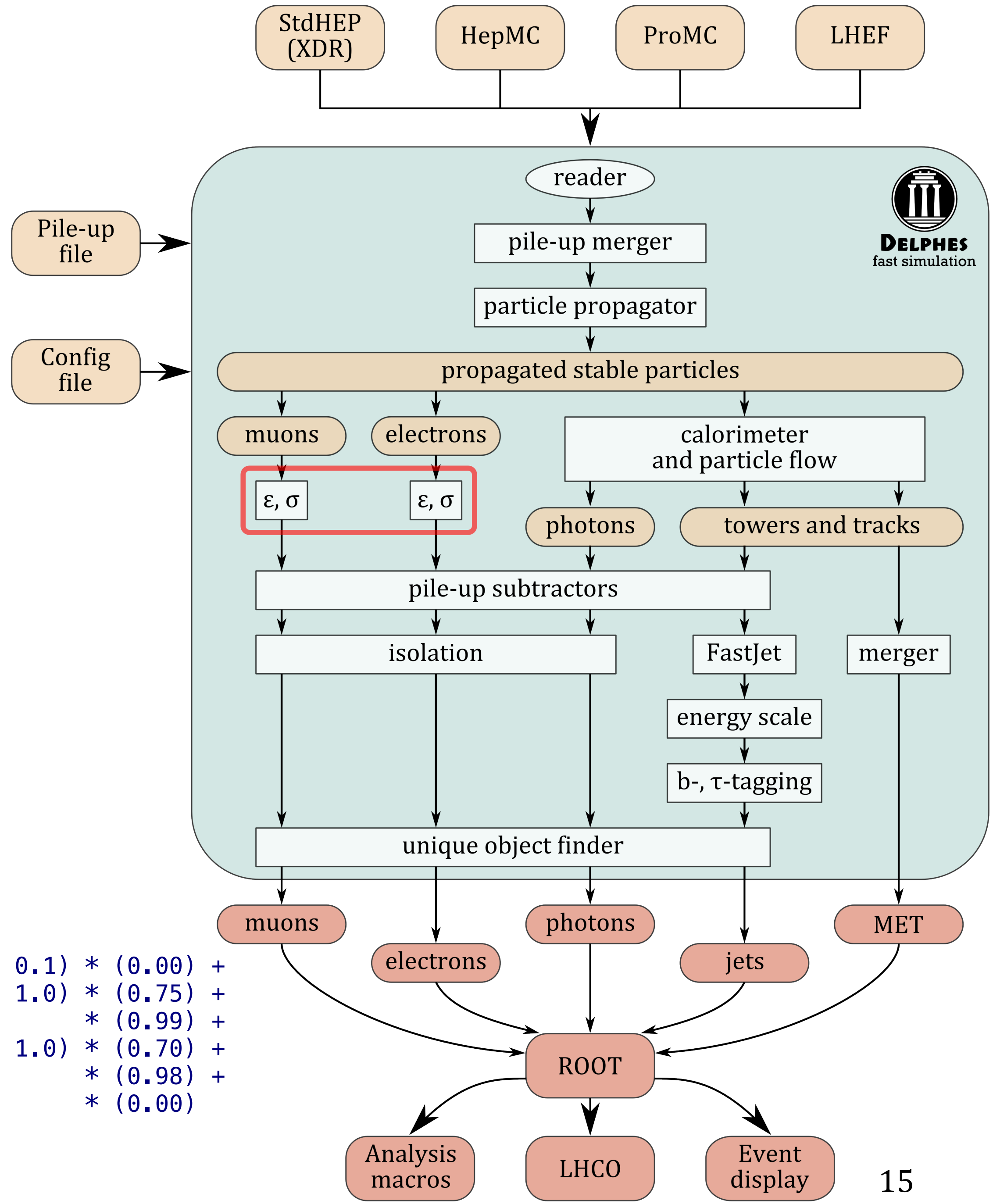


Tracking Efficiency and Resolution

- efficiency and resolution depend on
  - particle type (charged, photon, electron, muon)
  - transverse momentum
  - pseudo-rapidity

- not real tracking/vertexing:
  - no fake tracks/conversions
  - no dE/dx measurements

- for example, here is how the tracking efficiency for muons can be encoded:




```
# tracking efficiency formula for muons
set EfficiencyFormula {
                                                  (pt <= 0.1) * (0.00) +
                    (abs(eta) <= 1.5) * (pt > 0.1 && pt <= 1.0) * (0.75) +
                    (abs(eta) <= 1.5) * (pt > 1.0)              * (0.99) +
  (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 0.1 && pt <= 1.0) * (0.70) +
  (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 1.0)              * (0.98) +
                    (abs(eta)  > 2.5)                           * (0.00)
}
```

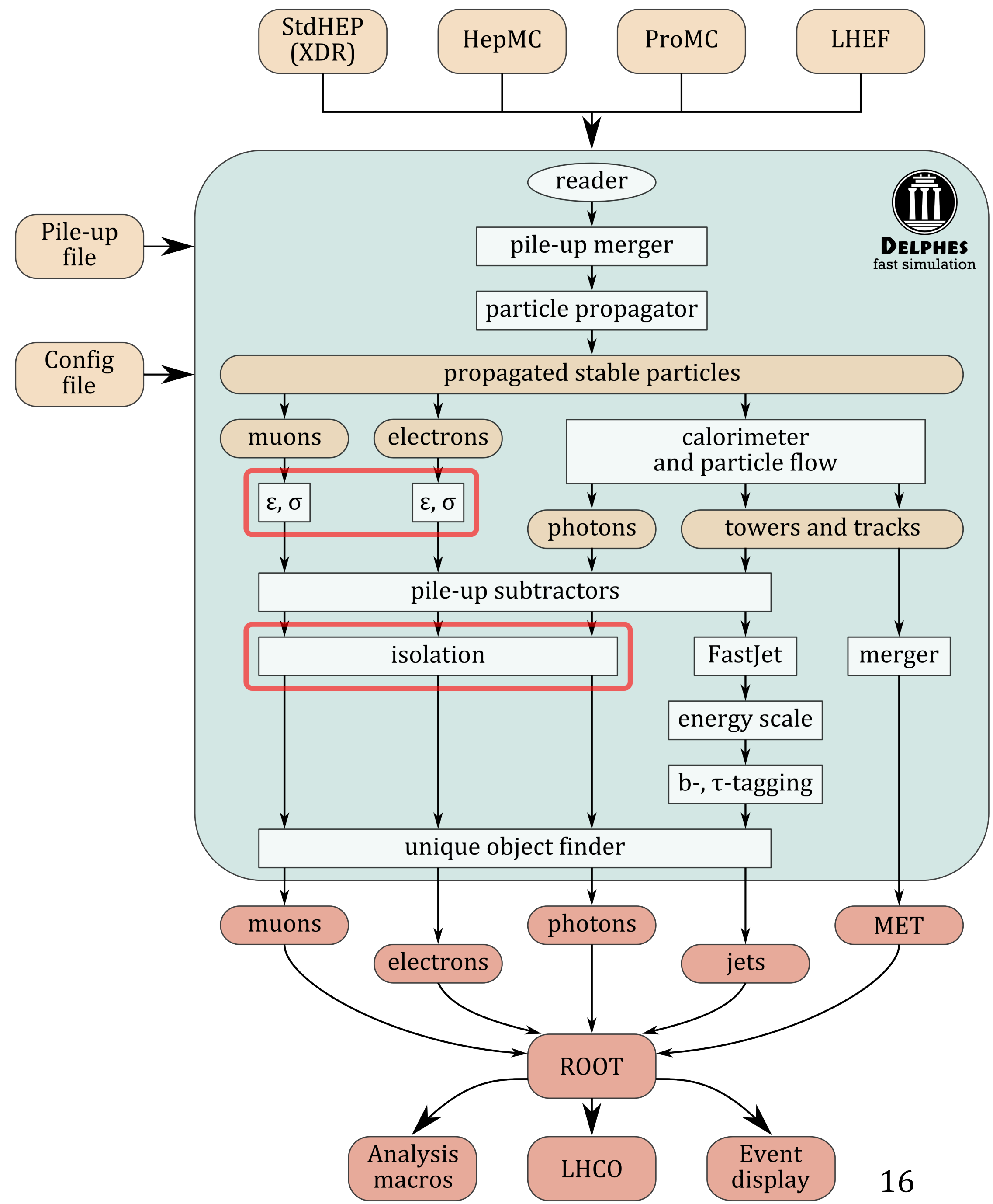
Photons, Electrons and Muons

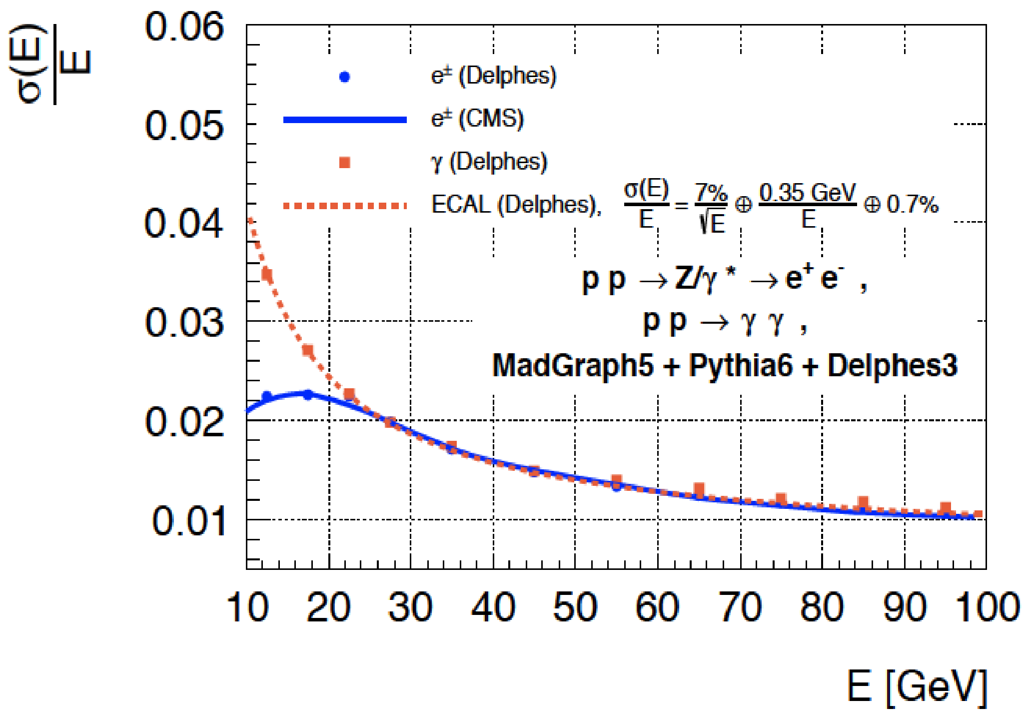
- identified via their PDG ID

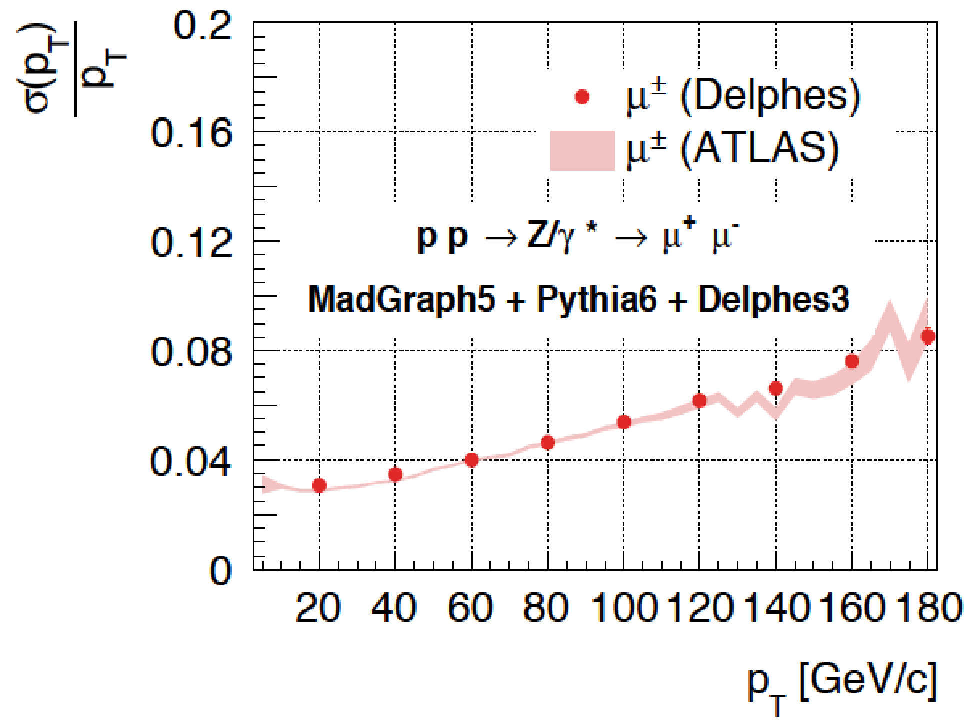
- **photons** are smeared according to the ECAL resolution
- **electrons** are smeared according to the tracker and ECAL resolution
- **muons** do not deposit energy in the calorimeter (independent smearing parametrized in p_T and η)

- modular structure allows to easily define various **isolation** criteria

- not implemented (yet):
  - misidentification,
  - punch-through,
  - brehmstrahlung,
  - conversions







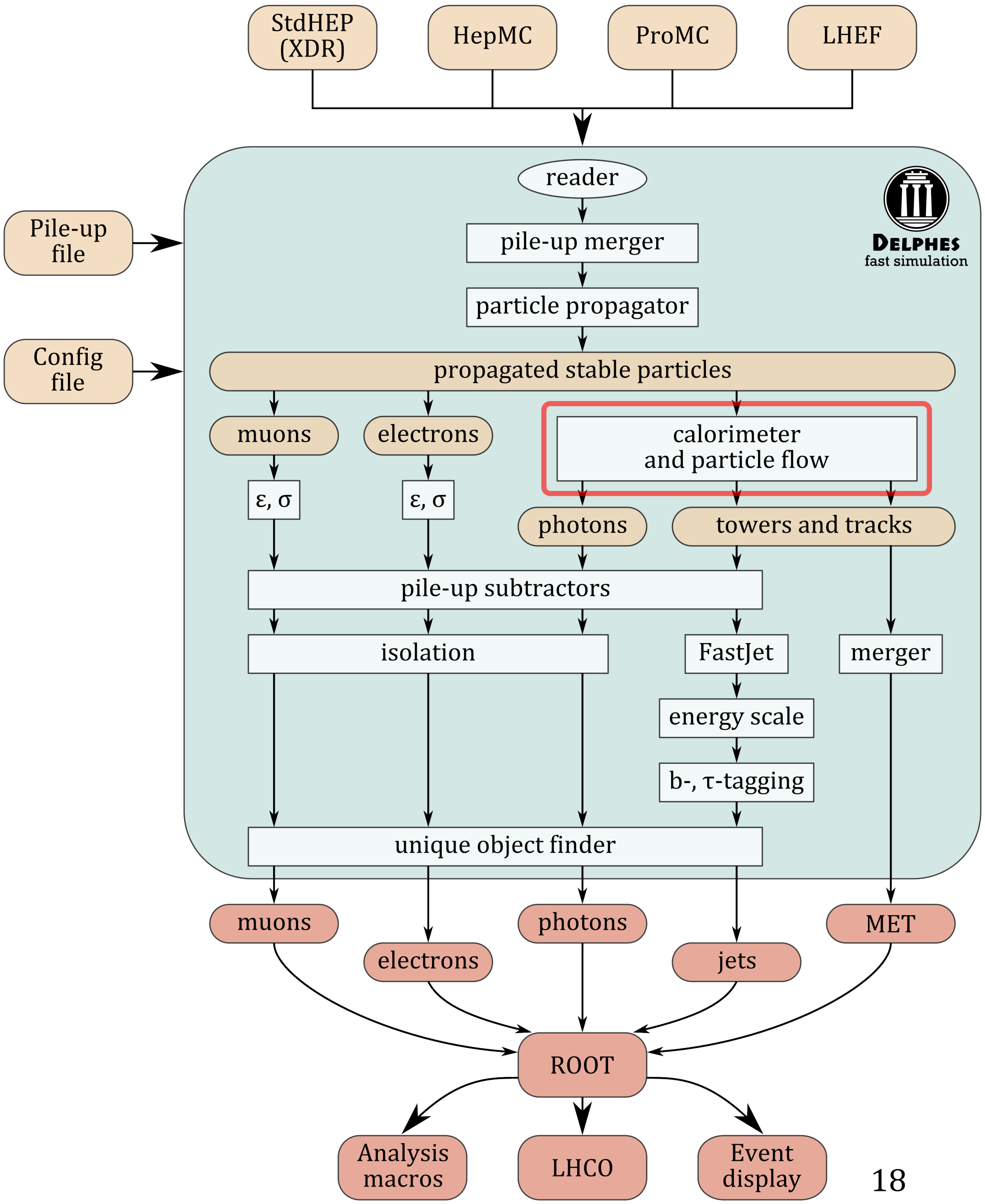
→ excellent agreement

Calorimetry

- ECAL/HCAL calorimeters have same **segmentation** in η and φ
- each particle that reaches the calorimeters **deposits a fraction of its energy** in one ECAL cell (f_{ECAL}) and one HCAL cell (f_{HCAL}):

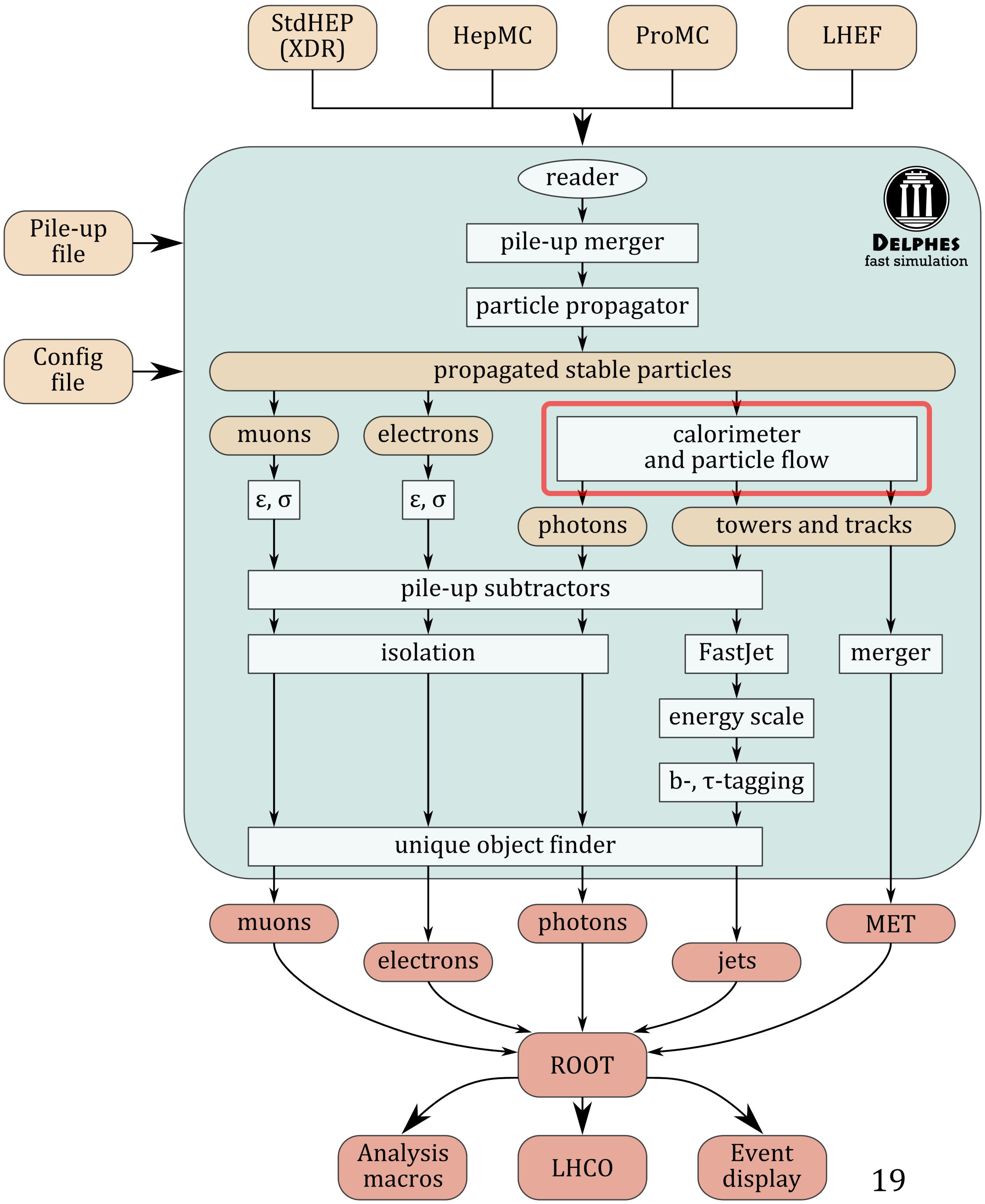
particles	f_{ECAL}	f_{HCAL}
$e\ \gamma\ \pi^0$	1	0
$K^0_S\ \Lambda^0$	0.3	0.7
$\nu\ \mu$	0	0
others	0	1

- particle energy is **smeared** according to the calorimeter region it reaches
- no energy sharing between the neighboring cells
- no longitudinal segmentation in the calorimeters



Particle Flow Emulation

- Delphes attempts to **reproduce** performance of the Particle Flow algorithm
- **tracking** and **calorimeter** information allows to reconstruct high resolution objects for later use (jets, missing E_T, H_T)
- assume σ(trk) < σ(calo)
- separate neutral and charged calorimeter deposits has crucial implications for pile-up subtraction



Particle Flow Emulation: Examples

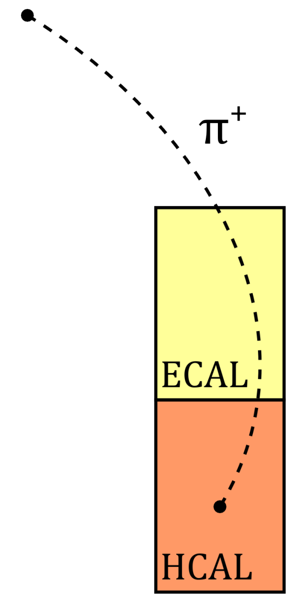
- Example 1:
  - pion of 10 GeV

    $E_{HCAL}(\pi^+) = 15$ GeV

    $E_{trk}(\pi^+) = 11$ GeV
  - Particle Flow algorithm creates:

    PF-track, with energy $E_{PF\text{-}trk} = 11$ GeV

    PF-tower, with energy $E_{PF\text{-}tower} = 4$ GeV



- Example 2:
  - pion of 10 GeV and photon of 20 GeV

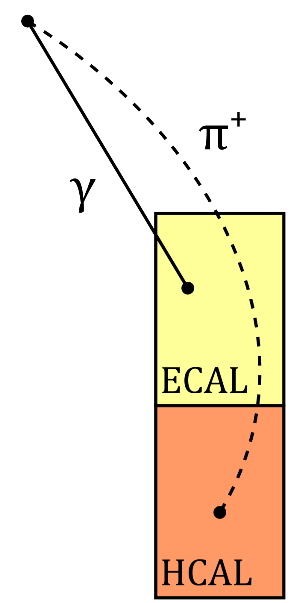
    $E_{ECAL}(\gamma) = 18$ GeV

    $E_{HCAL}(\pi^+) = 15$ GeV

    $E_{trk}(\pi^+) = 11$ GeV
  - Particle Flow algorithm creates:

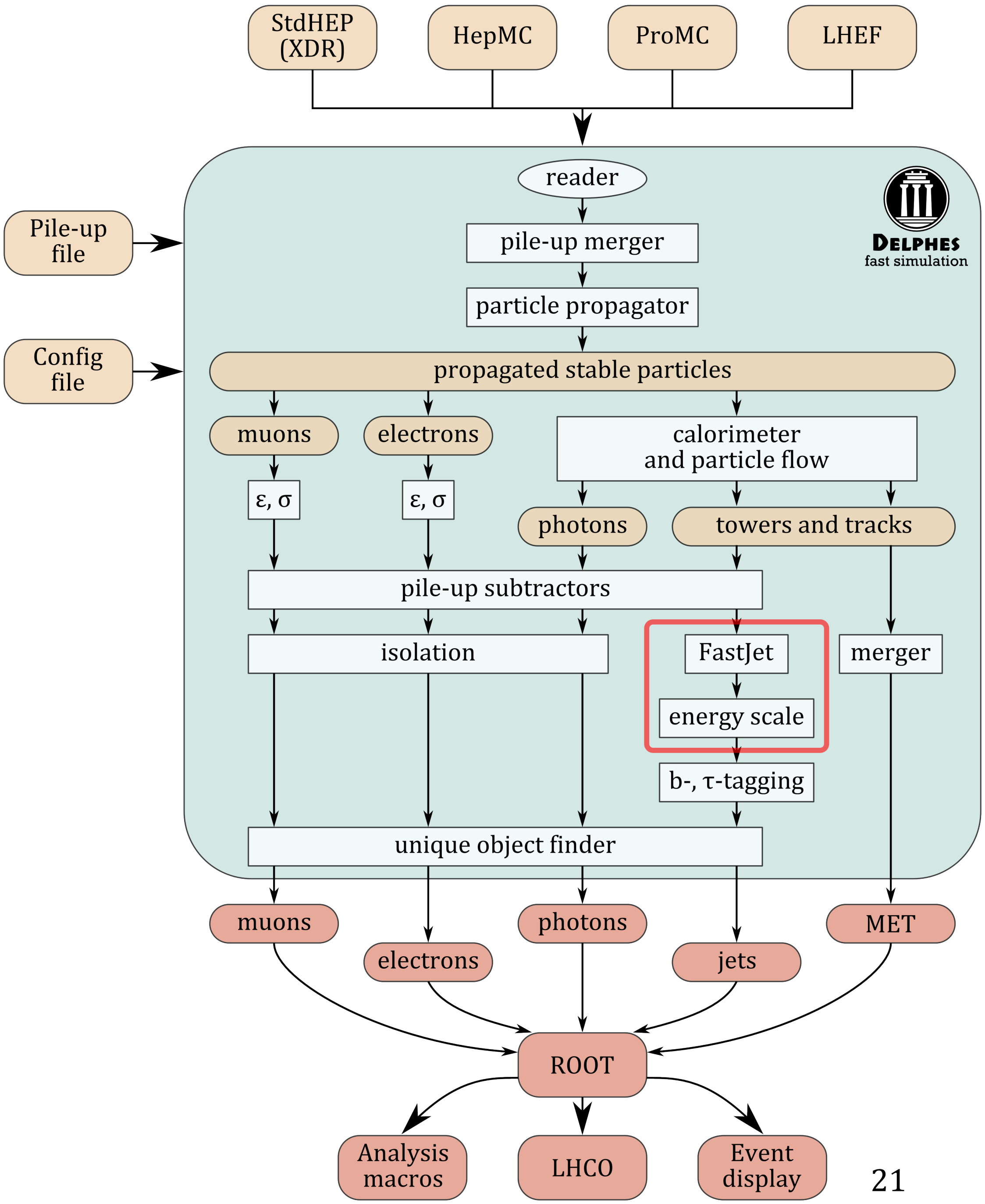
    PF-track, with energy $E_{PF\text{-}trk} = 11$ GeV

    PF-tower, with energy $E_{PF\text{-}tower} = 18 + 4$ GeV

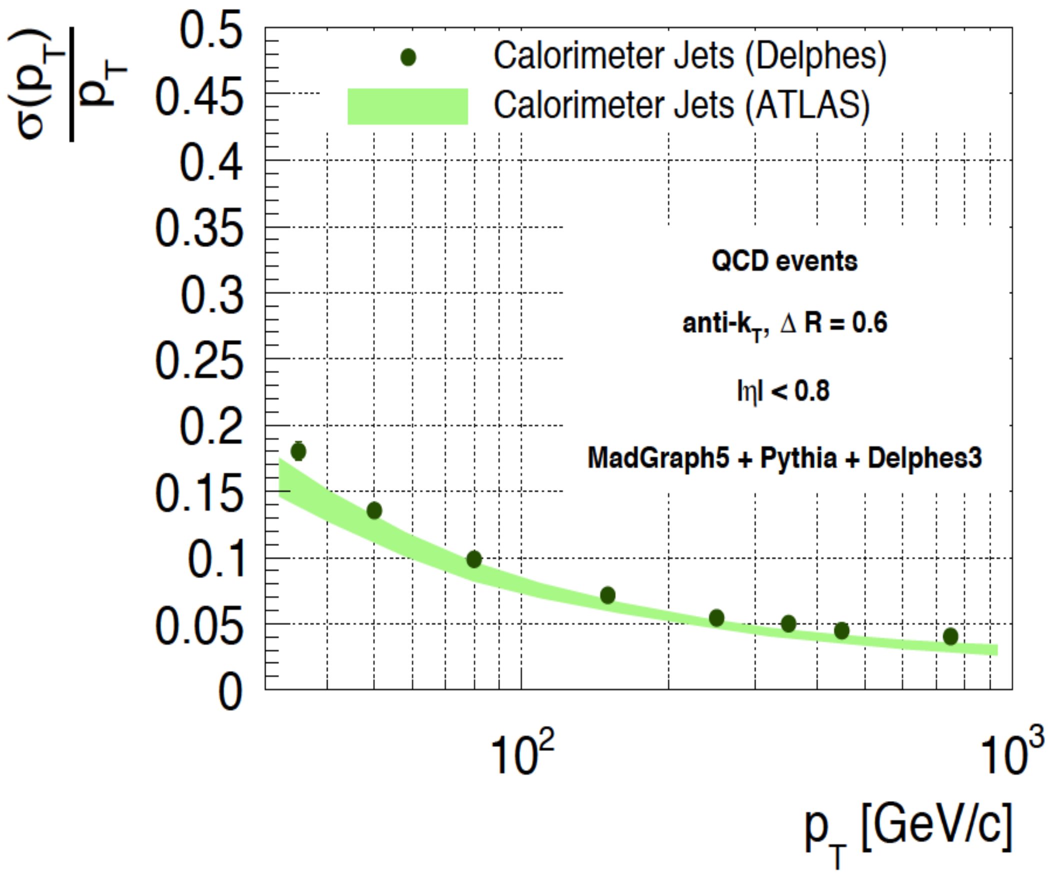


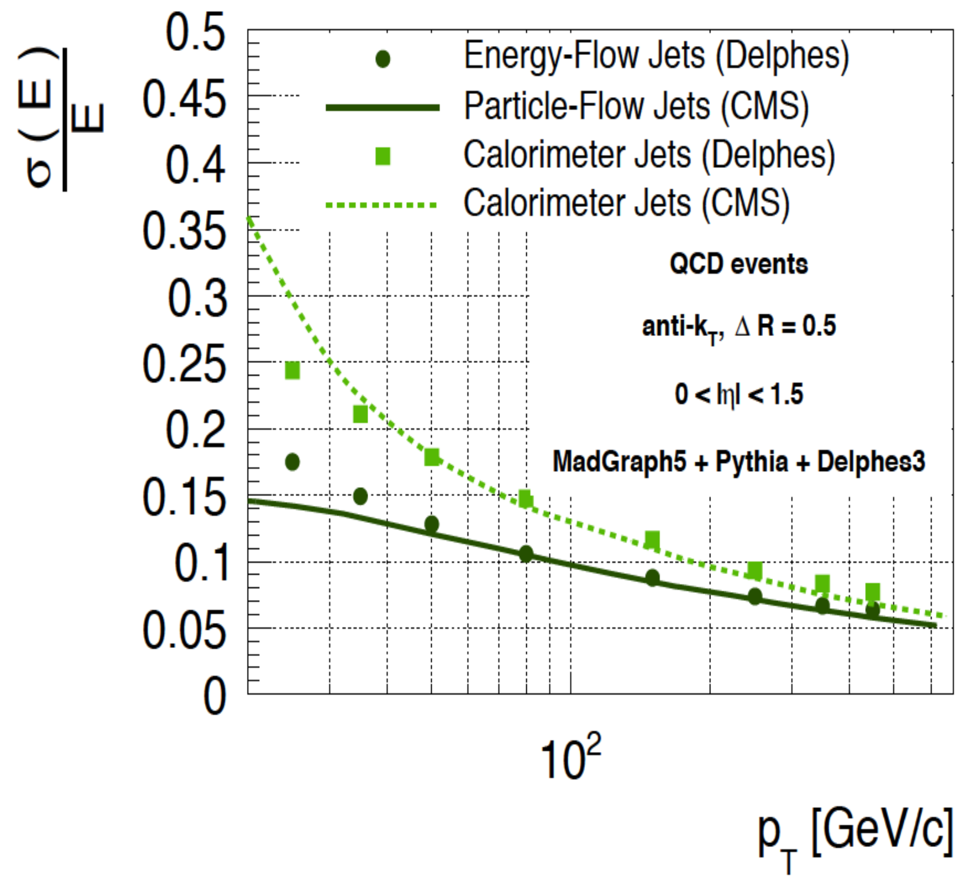
Jets

- Delphes uses the FastJet library [Eur.Phys.J. C72 (2012) 1896] to reconstruct jets
- wide set of algorithms available
- possible inputs:
  - particles produced by an event generator
  - calorimeter towers
  - Particle Flow objects
- jet energy scale corrections can be applied



Jets: Validation



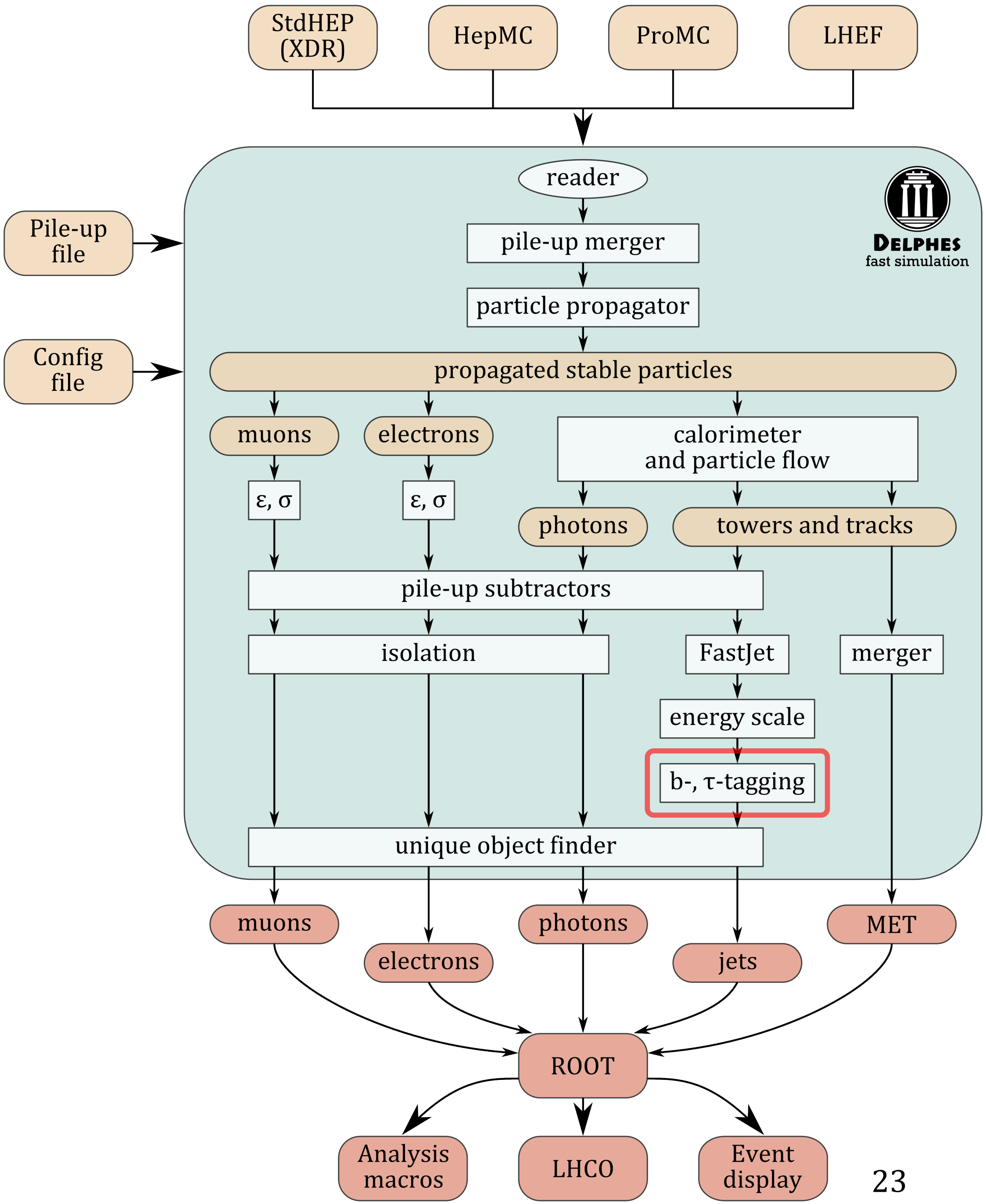


→ **good agreement**

b- and τ- jets

- **parametrized** b- (τ-) tagging
  - find the heaviest quark (or τ) within a cone of radius ΔR around the jet axis
  - apply corresponding efficiency or mistag rate

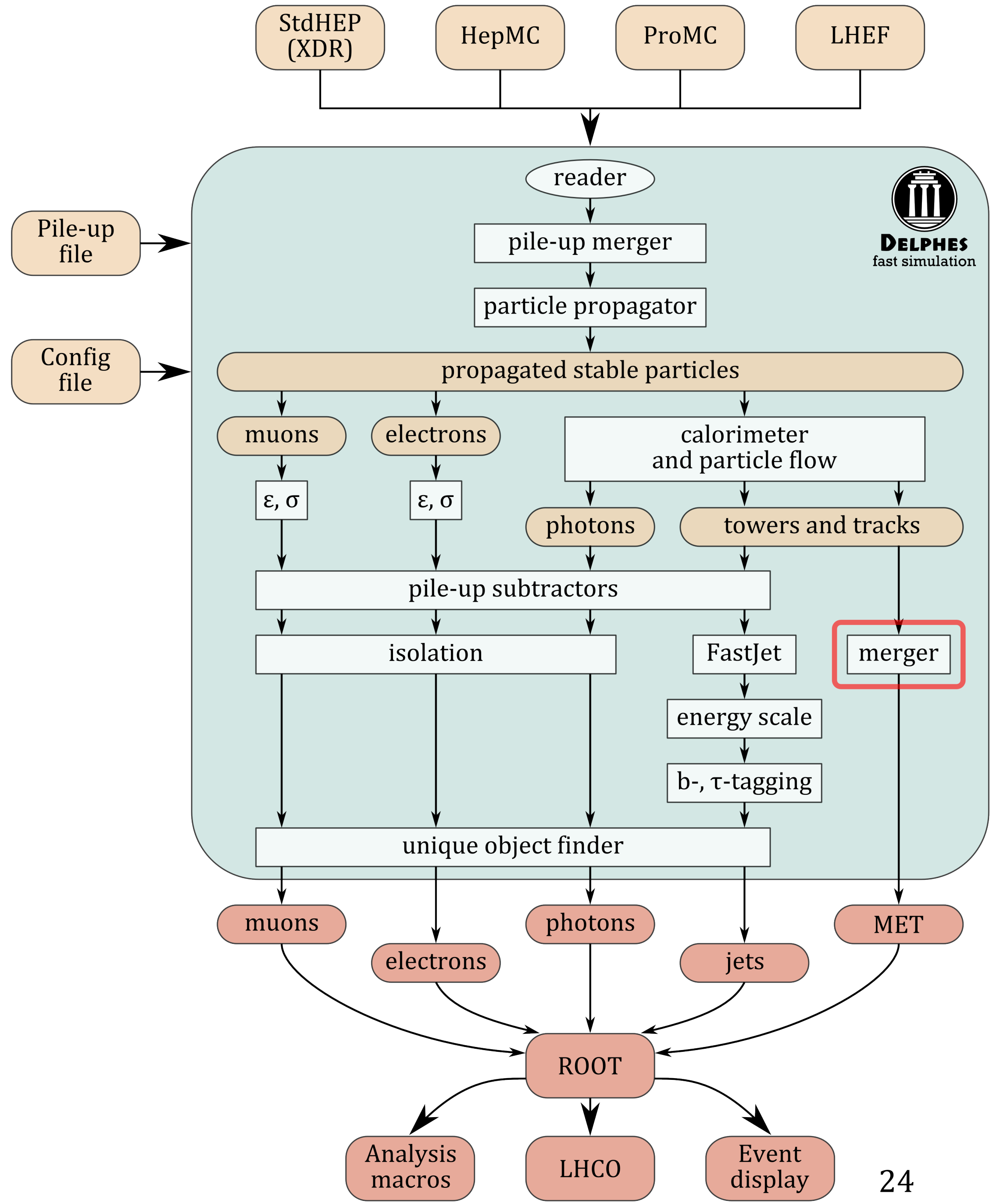
- **track counting** b-tagging
  - **count** tracks within jet with **large impact parameter significance**
  - apply a selection criterion

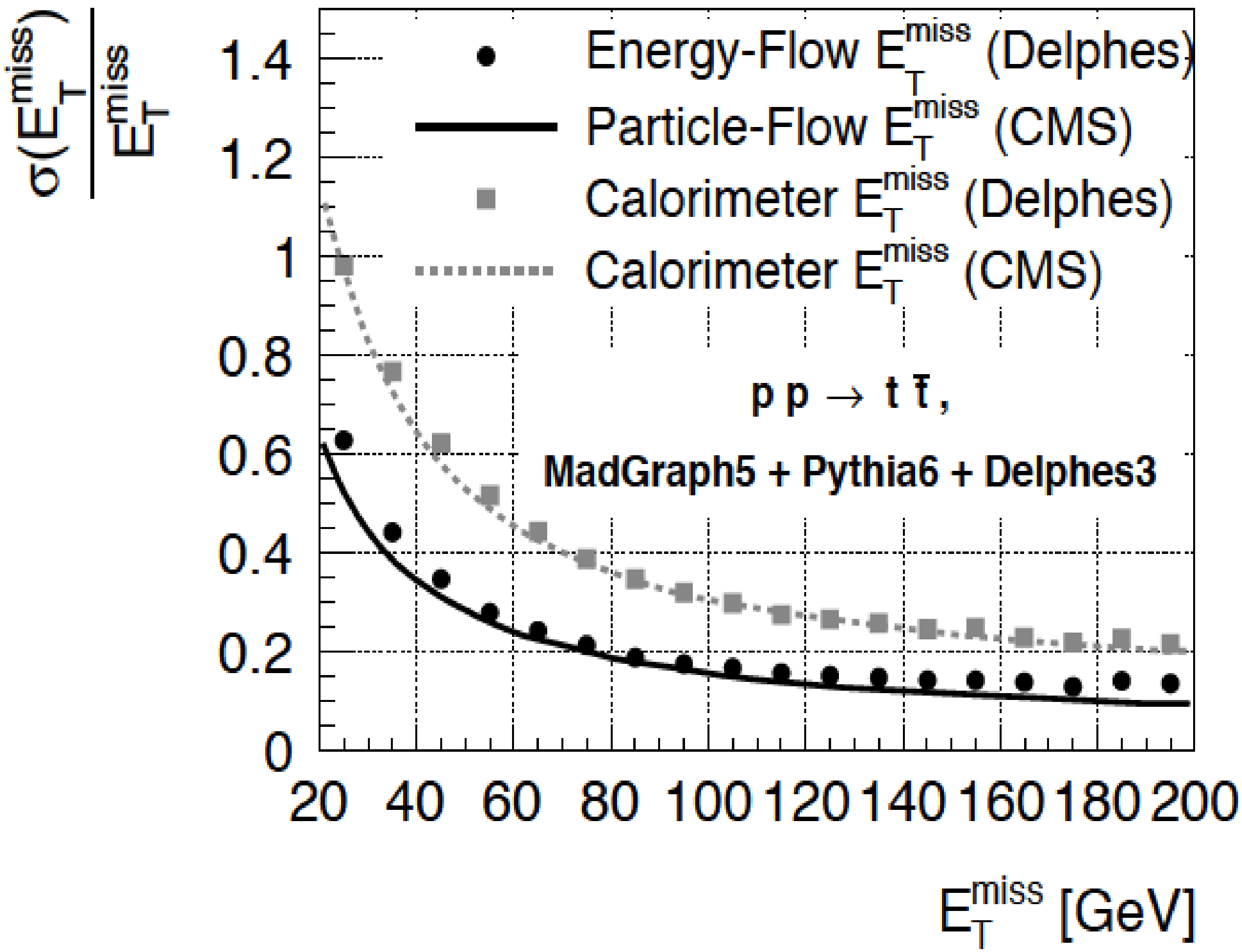


Missing E_T

- $$\overrightarrow{E_T}^{\text{miss}} = -\sum_i \overrightarrow{p_T}(i)$$

- possible inputs:
  - particles produced by an event generator
  - calorimeter towers
  - Particle Flow objects

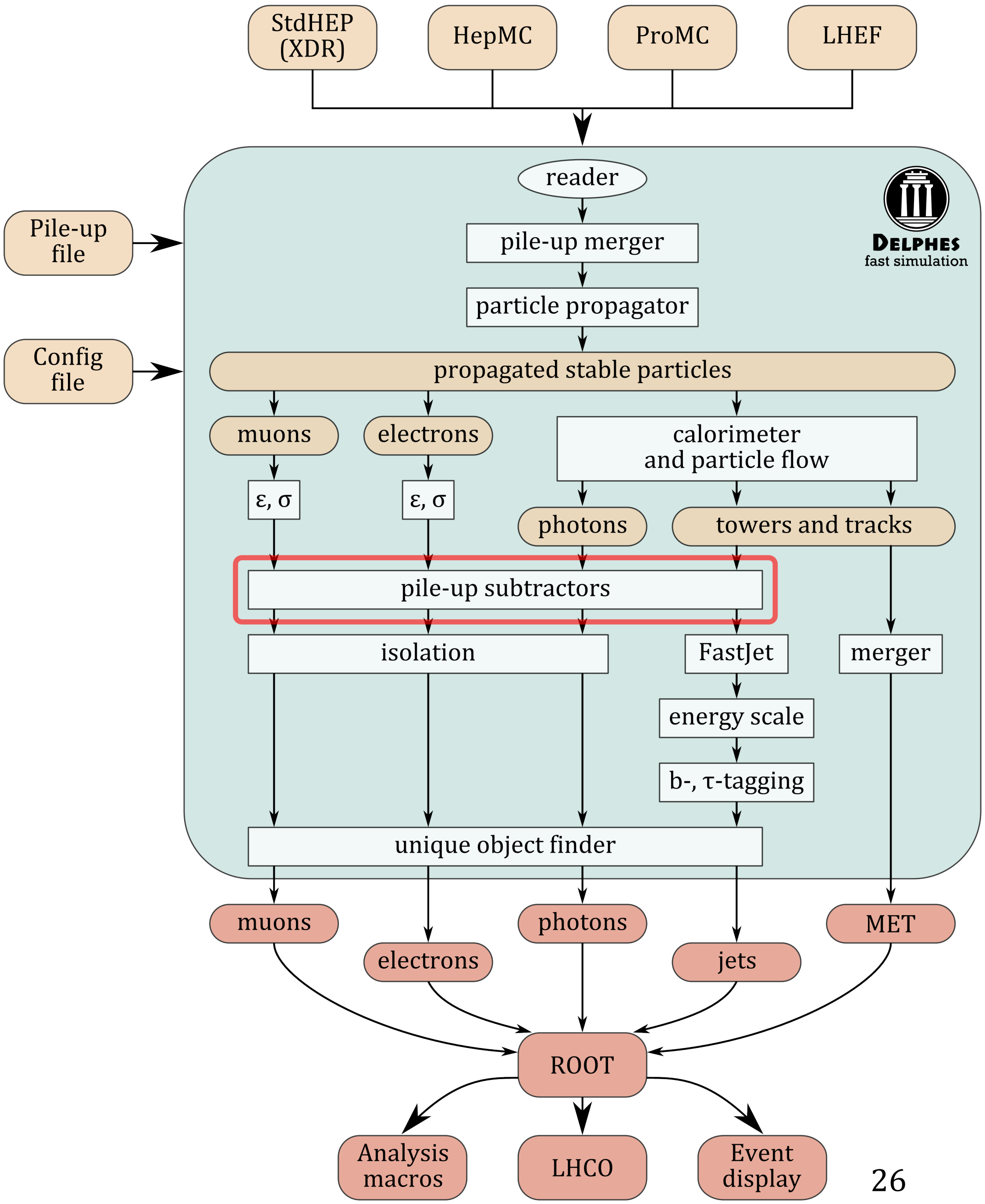




→ **excellent agreement**

Pile-up Subtraction

- **charged** pile-up subtraction (most effective if used with the particle flow algorithm)
  - remove all charged particles with $z_0 > |Z_{res}|$
- **residual** pile-up subtraction (needed for jets and isolation)
  - use FastJet to compute **pile-up density (ρ)** and **jet area (A)**
  - jet correction: $p_T \to p_T - \rho A$ (JetPileUpSubtractor module)
  - lepton isolation correction: $\sum p_T \to \sum p_T - \rho \pi R^2$ (Isolation module)
  - subtraction can be $|\eta|$ dependent



Pile-up Subtraction: Validation

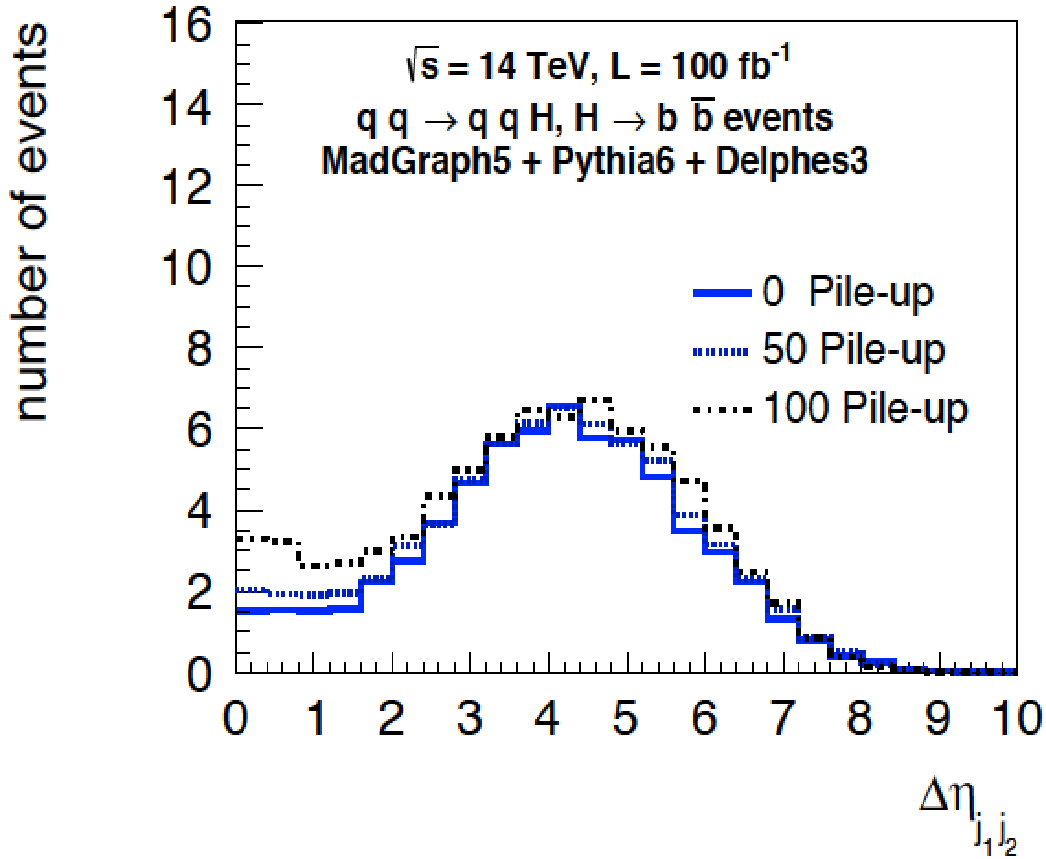
- $H \rightarrow bb$ in VBF channel expected to be highly affected by pile-up

- irreducible background: bb + jets

- select at least 4 jets with p_T > 80, 60, 40, 40 (at least 2 b-tagged jets, at least 2 light jets)

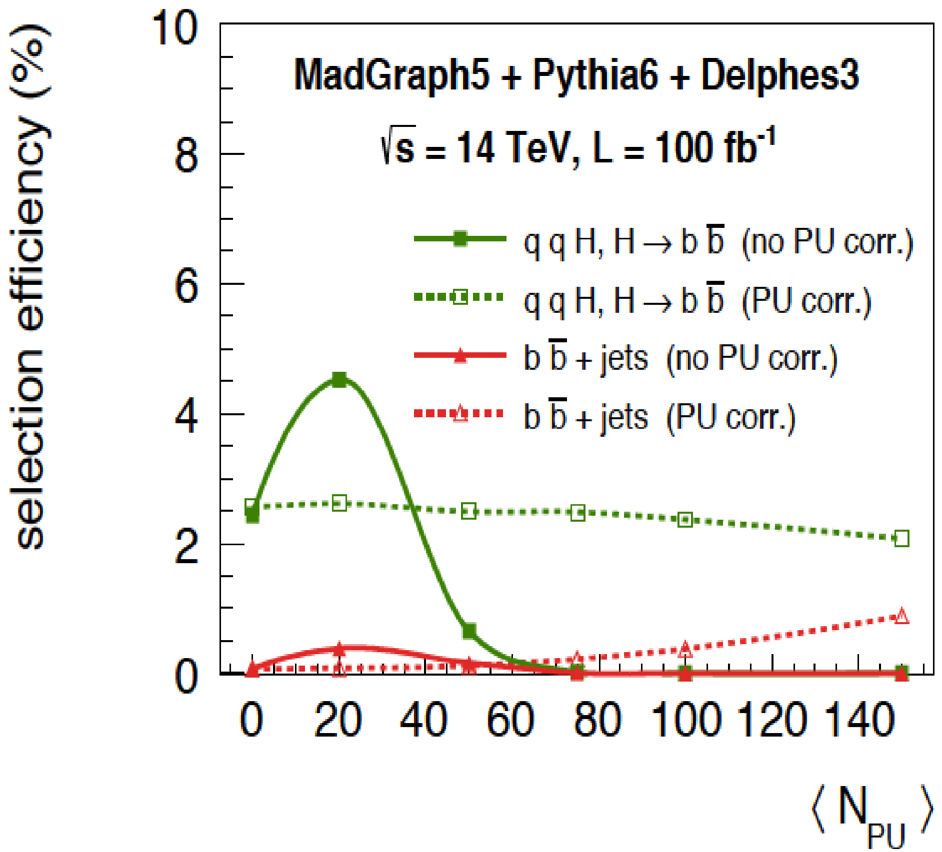
- emergence of pile-up jets in the central region:

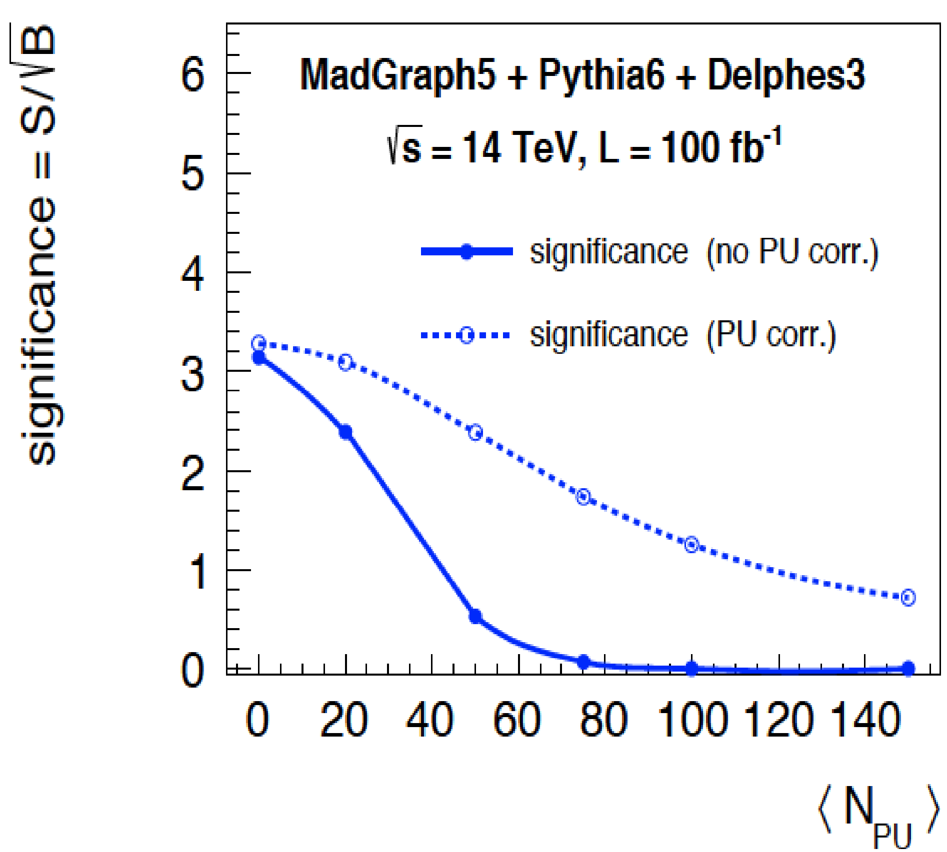
  → depletion of rapidity gap



Pile-up Subtraction: Validation

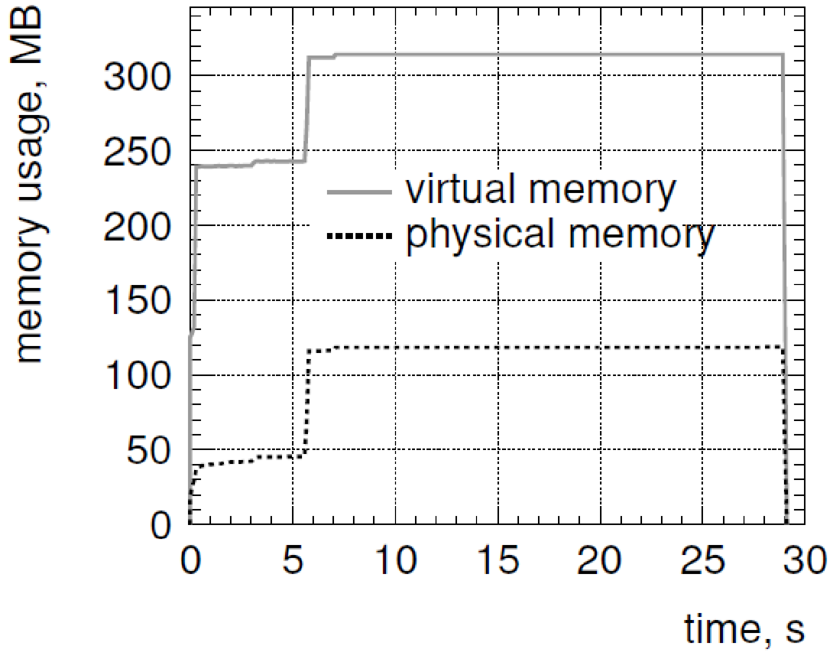
- require large rapidity gap between light jets, no hadronic activity in between
- 100 < m(bb) < 200 GeV

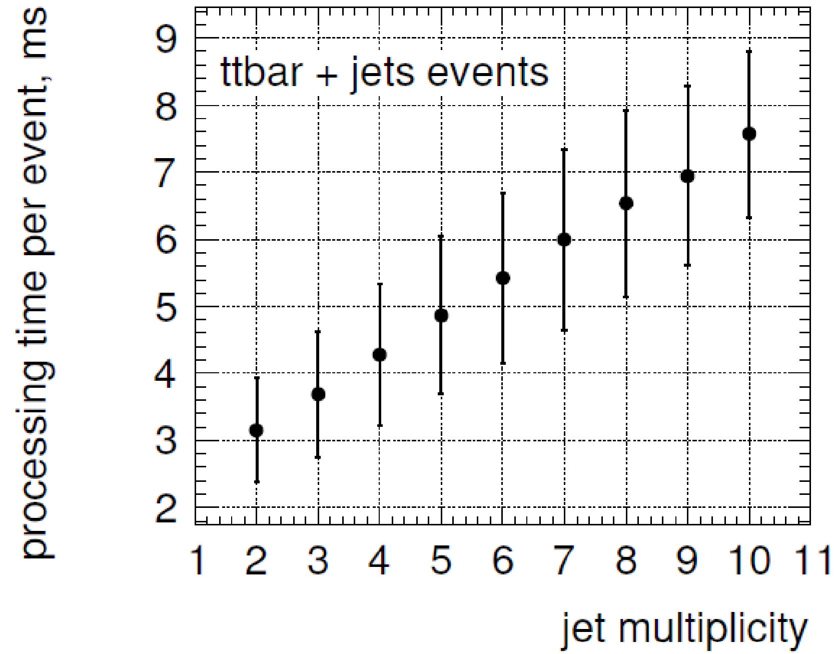




Performance, Analysis and Visualization

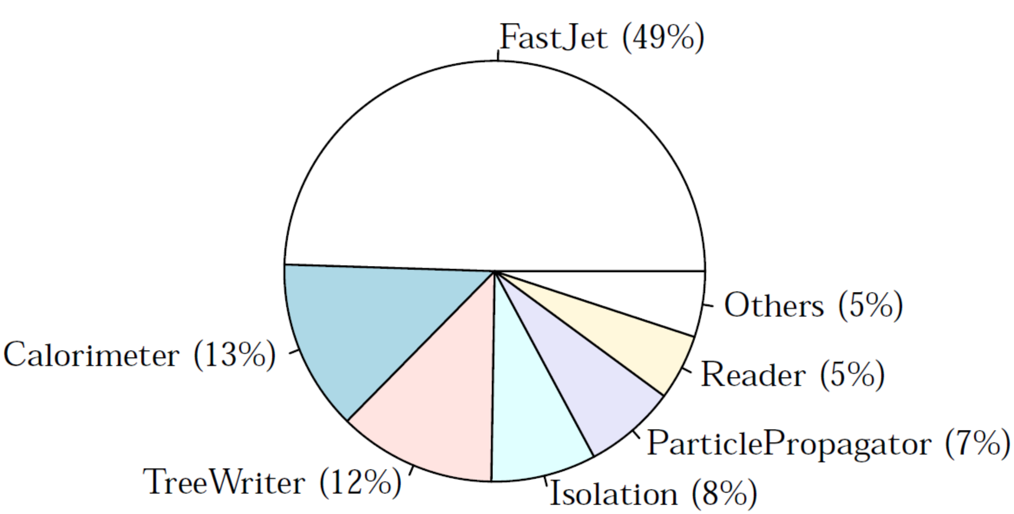
Performance



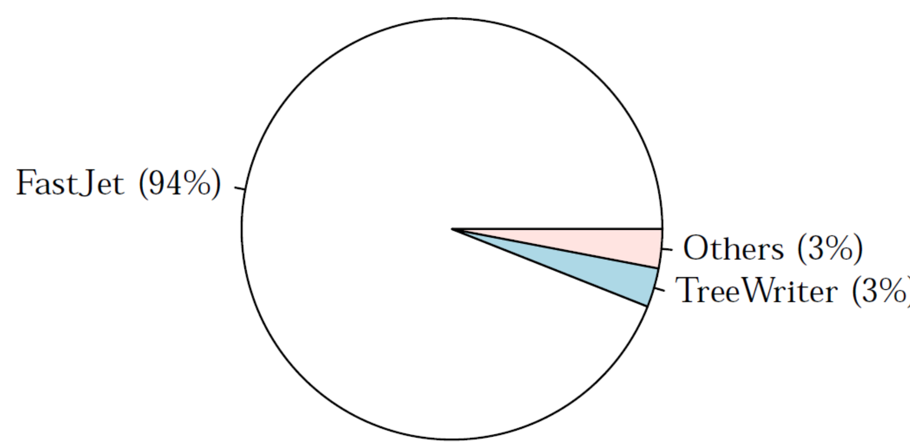


Relative CPU time used by the Delphes modules

0 pile−up



50 pile−up



Analysis

- Analyzing Delphes' output is simplified as much as possible, by providing intuitive tools:
  - ExRootAnalysis
    - C++/ROOT
    - helper classes for easier access to ROOT trees
  - DelphesAnalysis
    - Python/pyROOT
    - helper classes for event selection and control plots

- In both cases, examples are provided for immediate start

- No need to learn a big framework like in large experimental collaborations or to redo everything from scratch:
  - full analysis can be written in O(minutes) ~ O(hours)
  - tell what you want to see and get the histogram

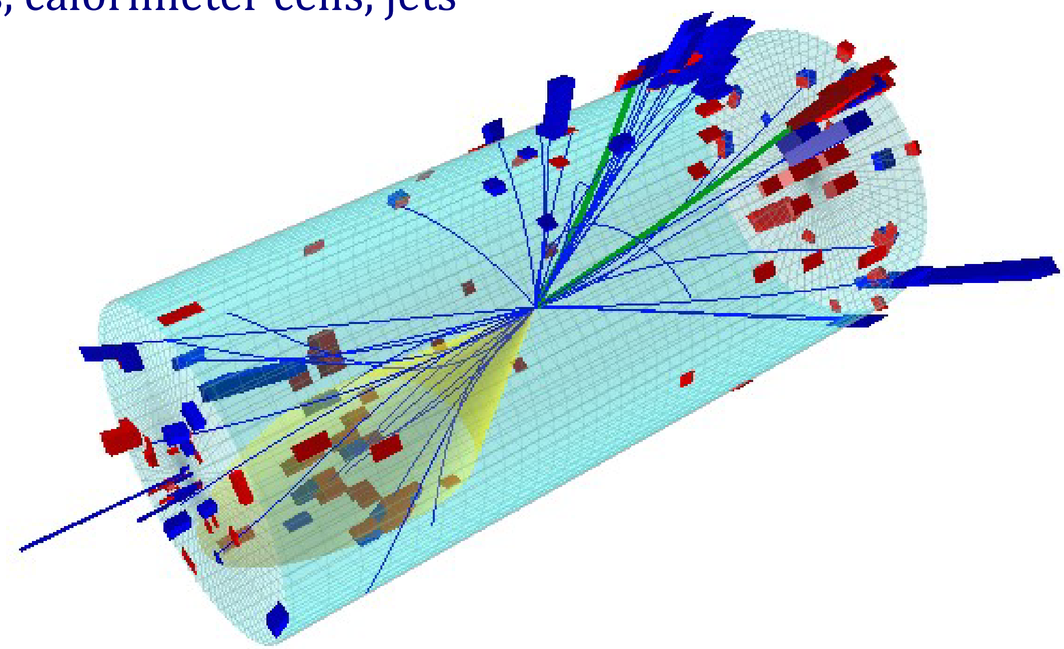
- Of course, you can use your favorite code… Delphes output is a standard ROOT tree…

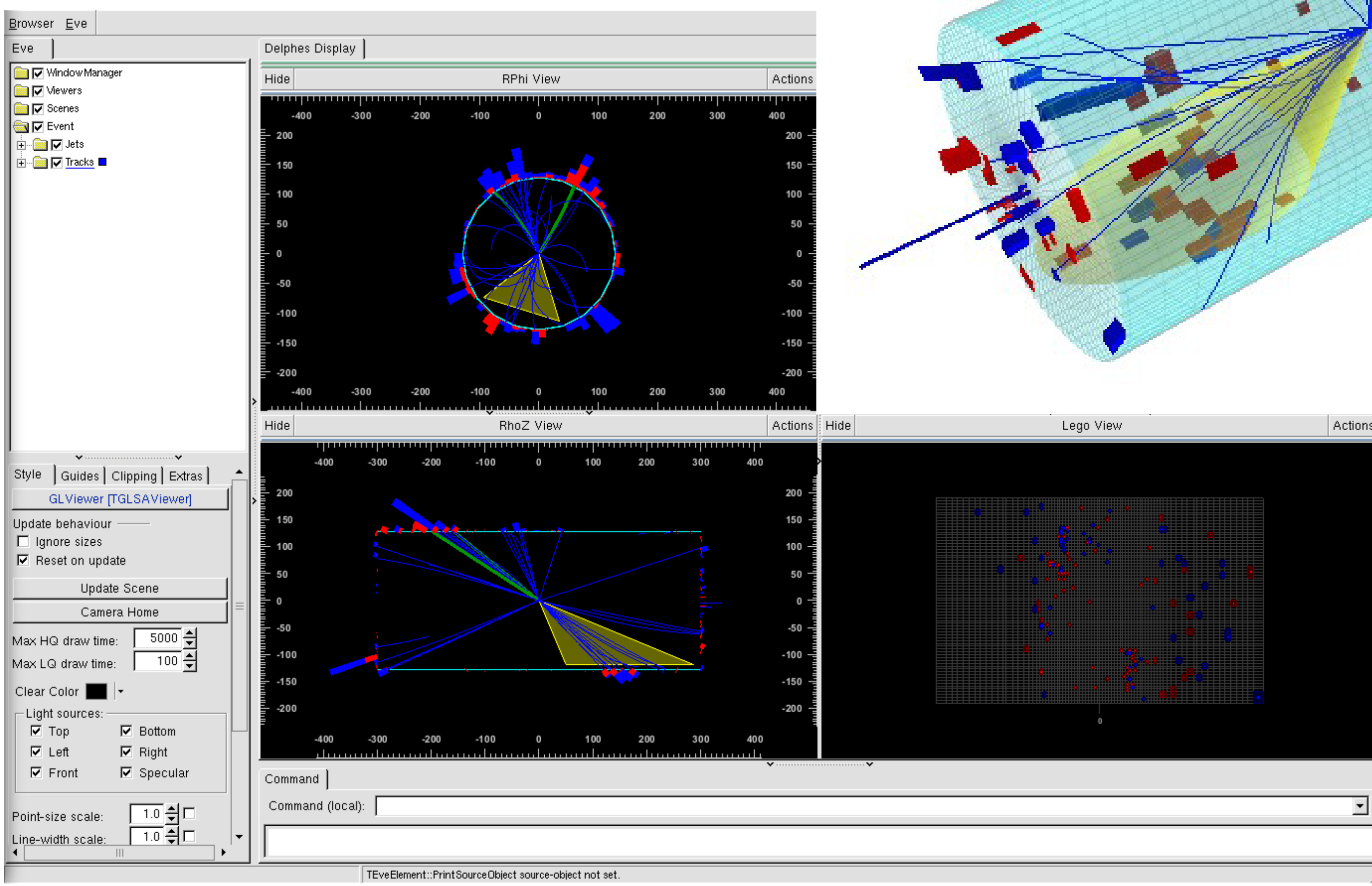
→ see the tutorial later this afternoon

Visualization

A basic event display is provided, based on ROOT EVE

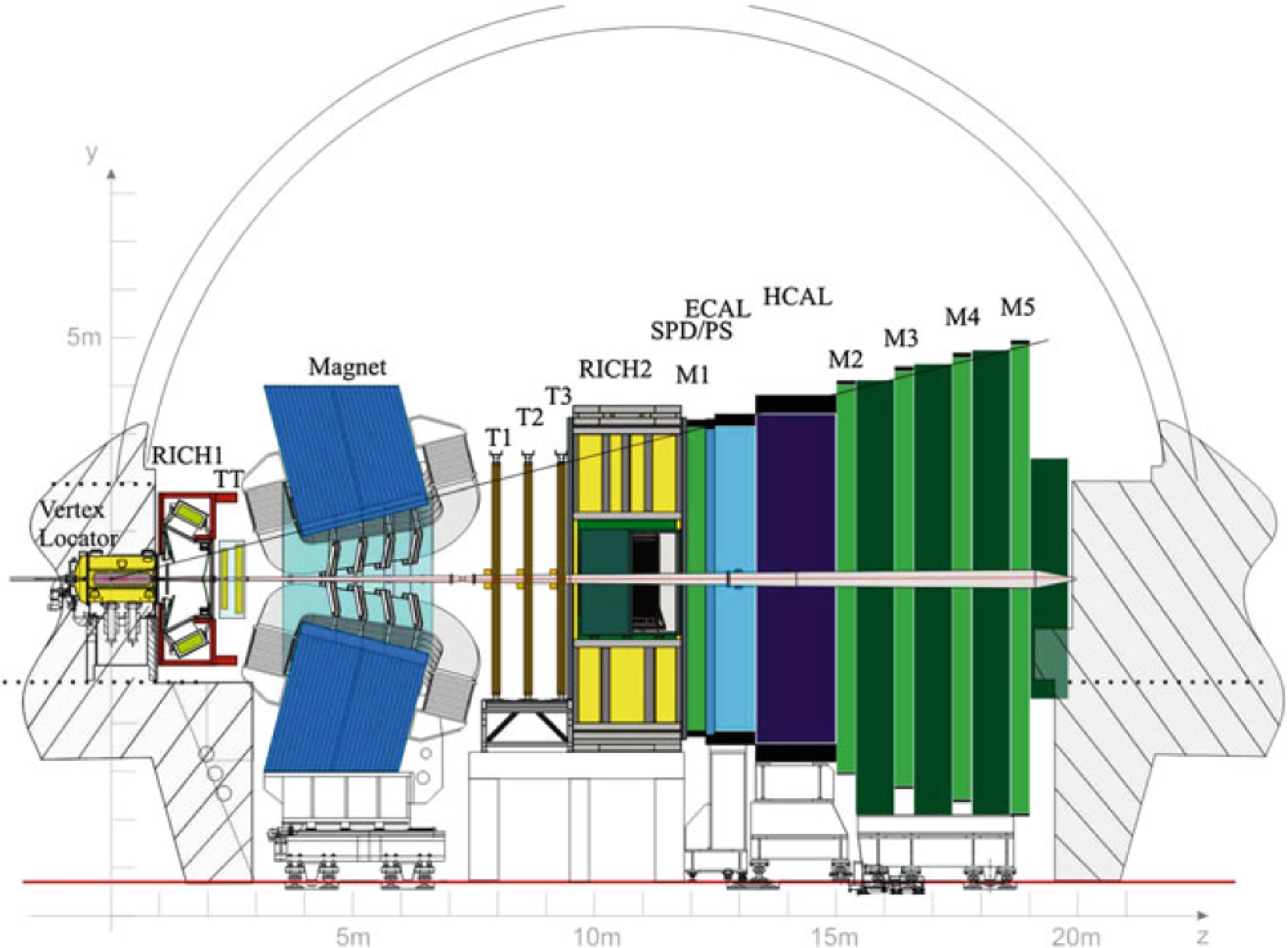
- displays tracks, electrons, muons, calorimeter cells, jets
- more detailed version planed





How to Simulate LHCb and AFTER@LHC?

LHCb Layout



y
5m
Magnet
RICH1
TT
Vertex
Locator
T1
T2
T3
RICH2
M1
SPD/PS
ECAL
HCAL
M2
M3
M4
M5
5m
10m
15m
20m
z

Questions

- Output collections (γ, e, μ, π, K, p, jets, ???)
- Jet input collections
- Calorimeters or parametrized resolution σ(PID, E, θ, φ, ???)
- Particle propagation (parabolic trajectories?)
- Magnetic field (map or parametrization?)
- Vertexes
- ???

UCL Université catholique de Louvain

?

Final Remarks and Conclusions

When and When Not Delphes?

- When to use Delphes?
  - more advanced than parton-level studies
  - testing analysis methods (multivariate/Matrix Element)
  - test your model (CheckMATE)
  - scan big parameter space (SUSY-like)
  - preliminary tests of new geometries/resolutions (upgrades, Snowmass)
  - educational purpose (bachelor/master thesis)

- When not to use Delphes?
  - high precision studies
  - very exotic topologies (heavy stable charged particles)
  - study is sensitive to tails

Conclusions

- **Delphes 3** has been out for more than one year now, with **major improvements**:
  - modularity
  - pile-up implementation
  - revamped particle flow emulation
  - visualization tool based on ROOT EVE
  - example configuration files giving results on par with published performance of the LHC experiments (ATLAS and CMS)
  - fully integrated within MadGraph5

- To-do:
  - energy sharing between the neighboring calorimeter cells
  - longitudinal segmentation in the calorimeters
  - understand how to simulate LHCb and AFTER@LHC

People

Jerome de Favereau

Christophe Delaere

Pavel Demin

Andrea Giammanco

Vincent Lemaitre

Alexandre Mertens

Michele Selvaggi

the community…