DELPHES
fast simulation

Michele Selvaggi, for the Delphes Team

Université catholique de Louvain (UCL)
Center for Particle Physics and Phenomenology (CP3)

JHEP 02 (2014) 057

SLAC – 100 TeV workshop
23 April 2014

Outline

- The Delphes Project
- Event Reconstruction
- New Features
- Delphes and hh@100TeV
- Conclusion

The Delphes Project

The Delphes project: A bit of history

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

- Since 2009, its development is **community-based**
  - **ticketing system** for improvement and bug-fixes
    → user proposed patches
  - **Quality control** and **core development** is done at the UCL

- In 2013, **DELPHES 3** was released:
  - modular software
  - new features
  - also included in MadGraph suite

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

- Website and manual: https://cp3.irmp.ucl.ac.be/projects/delphes

- Paper: JHEP 02 (2014) 057

The Delphes project: I/O and configurations

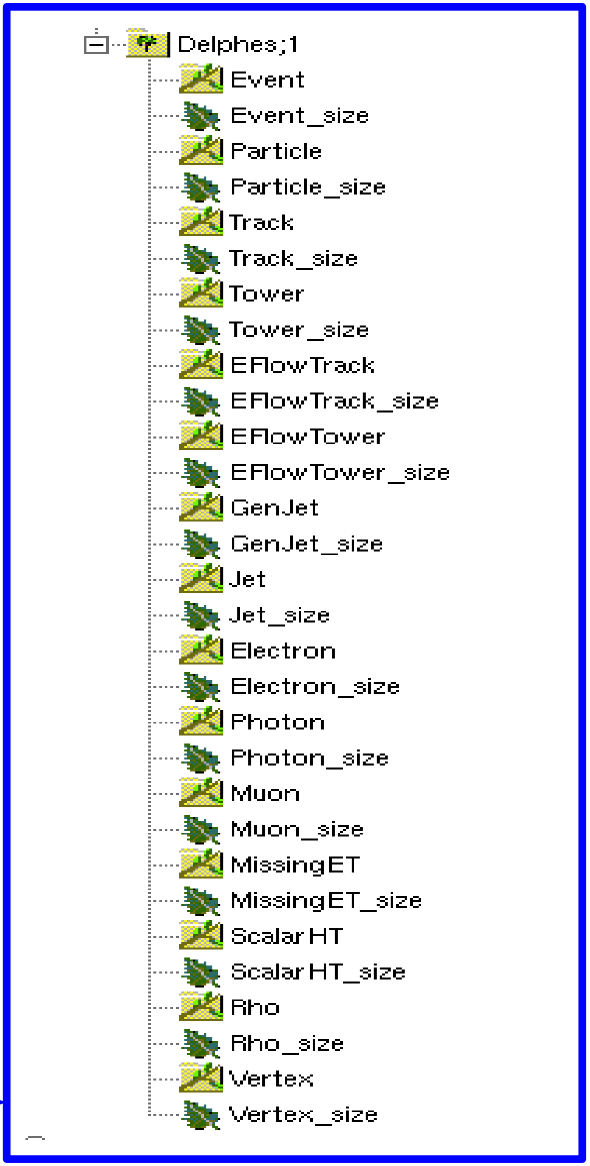
- **modular** C++ code

- Uses
  - ROOT classes [Comp. Phys. C. 180 (2009) 2499]
  - FastJet package [Eur. Phys. J. C 72 (2012) 1896]

- Input
  - Pythia/Herwig output (HepMC,STDHEP)
  - LHE (MadGraph/MadEvent)
  - ProMC

- Configuration file
  - Define geometry
  - Resolution/reconstruction/selection criteria
  - Output object collections

- Output
  - ROOT trees



default **CMS/ATLAS** and “dummy” future collider configurations are included

Detector simulation

- Full simulation (GEANT):

    - **simulates** particle-matter interaction (including e.m. showering, nuclear int., brehmstrahlung, photon conversions, etc ...) → 10 s /ev

- Experiment Fast simulation (ATLAS, CMS ...):

    - **simplifies** and makes faster simulation and reconstruction → 1 s /ev

- Parametric simulation:

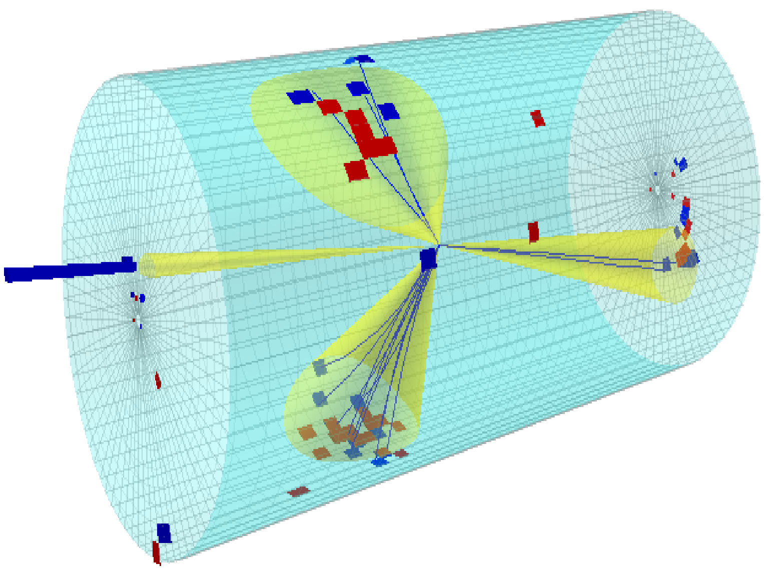
  Delphes, PGS**:**

    - **parameterize** detector response, reconstruct complex objects → 10 ms /ev

The Delphes project: Delphes in a nutshell

- **Delphes** is a **modular framework** that simulates of the response of a multipurpose detector in a **parameterized** fashion

- **Includes**:
  - pile-up
  - charged particle **propagation** in magnetic field
  - electromagnetic and hadronic **calorimeters**
  - **muon** system



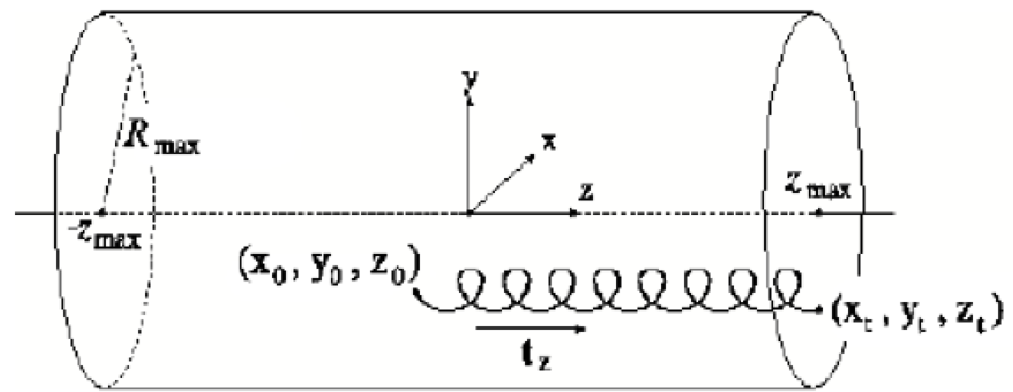
- **Provides**:
  - leptons (electrons and muons)
  - photons
  - jets and missing transverse energy (particle-flow)
  - taus and b's

The modules: Particle Propagation

- **Charged** and **neutral** particles are propagated in the magnetic field until they reach the calorimeters

- Propagation parameters:

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



- Efficiency/resolution depends on:

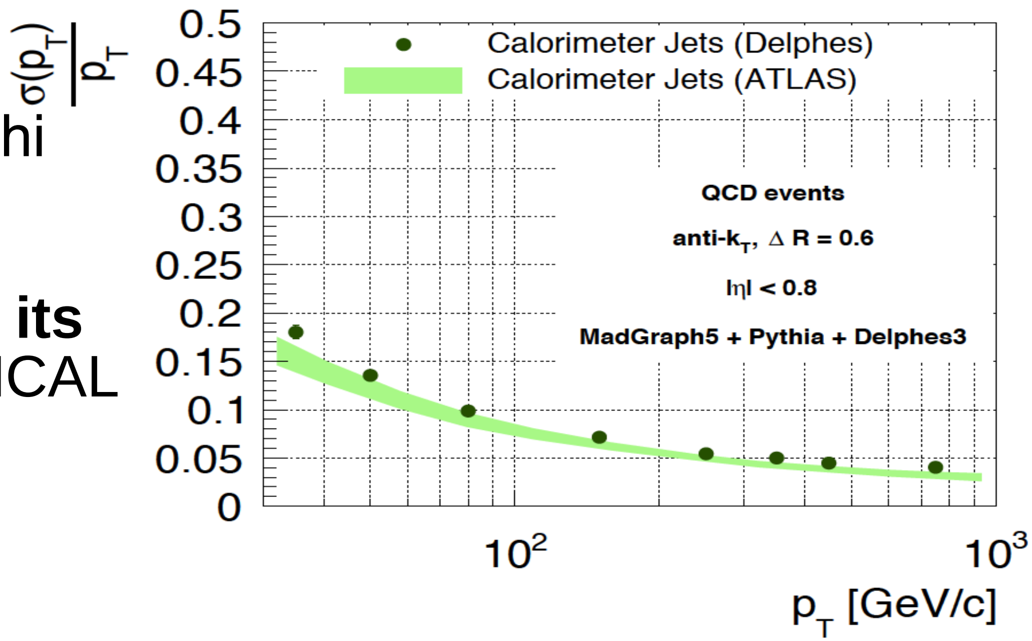
  - particle ID
  - transverse momentum
  - pseudorapidity

```
# efficiency formula for muons
add EfficiencyFormula {13} {                                          (pt <= 0.1)   * (0.000) + \
                                       (abs(eta) <= 1.5) * (pt > 0.1  && pt <= 1.0)   * (0.750) + \
                                       (abs(eta) <= 1.5) * (pt > 1.0)                 * (1.000) + \
                     (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 0.1  && pt <= 1.0)   * (0.700) + \
                     (abs(eta) > 1.5 && abs(eta) <= 2.5) * (pt > 1.0)                 * (0.975) + \
                     (abs(eta) > 2.5)                                                 * (0.000)}
```

No real tracking/vertexing !!
→ no fake tracks/ conversions (but can be implemented)
→ no dE/dx measurements

The modules: Calorimetry

- Can specify separate ECAL/HCAL **segmentation** in eta/phi
- Each particle that reaches the calorimeters **deposits a fraction of its energy** in one ECAL cell (f_{EM}) and HCAL cell (f_{HAD}), depending on its type:



particles	f_{EM}	f_{HAD}
e γ π^0	1	0
Long-lived neutral hadrons (K^0_s , Λ^0)	0.3	0.7
ν μ	0	0
others	0	1

- Particle energy is **smeared** according to the calorimeter cell it reaches

No Energy sharing between the neighboring cells
No longitudinal segmentation in the different calorimeters

The modules: Particle-Flow Emulation

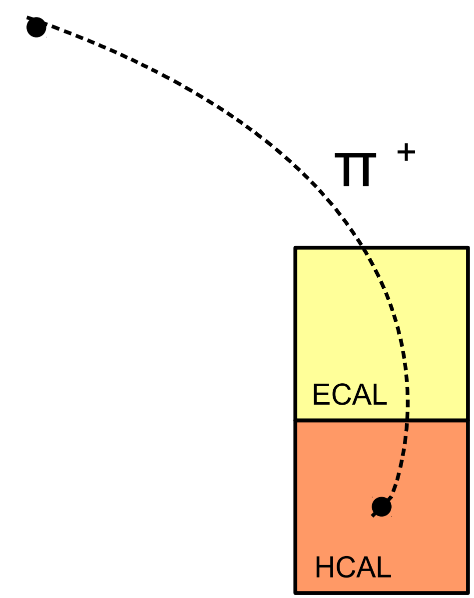
- Idea: Reproduce realistically the performances of the Particle-Flow algorithm.

- In practice, in DELPHES use **tracking and calo** info to reconstruct high reso. input objects for later use (jets, E_T^{miss}, H_T)

→ assume **σ(trk) < σ(calo)**

Example: A 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

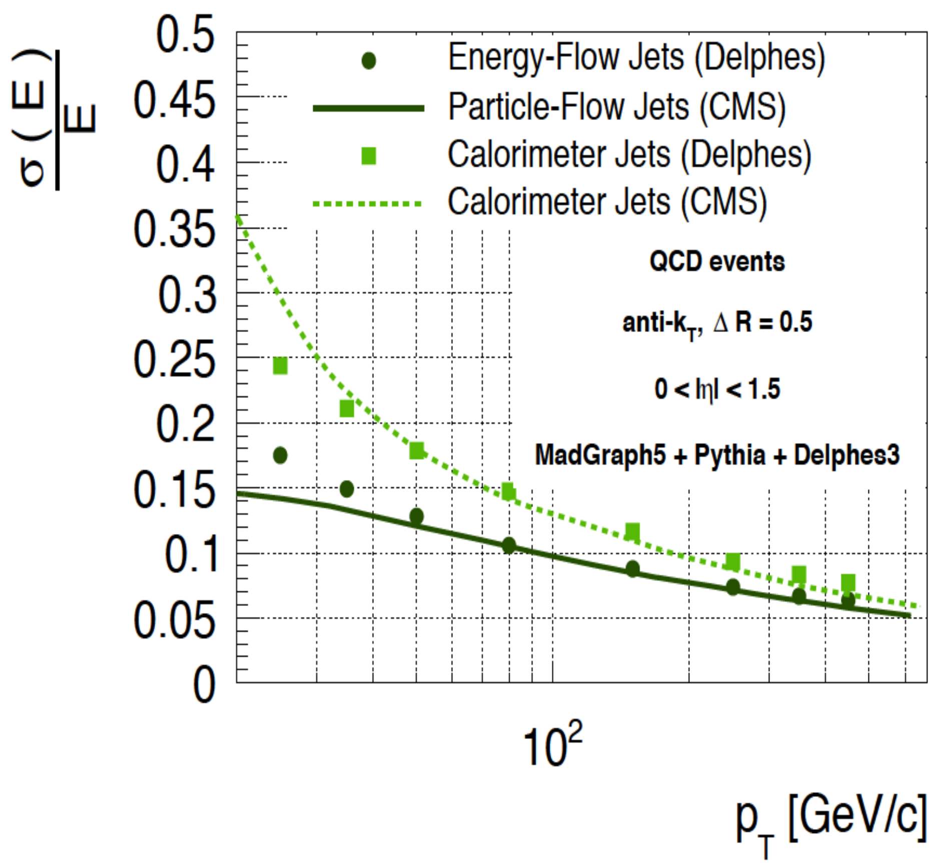
Separate neutral and charged calo deposits has crucial implications for pile-up subtraction

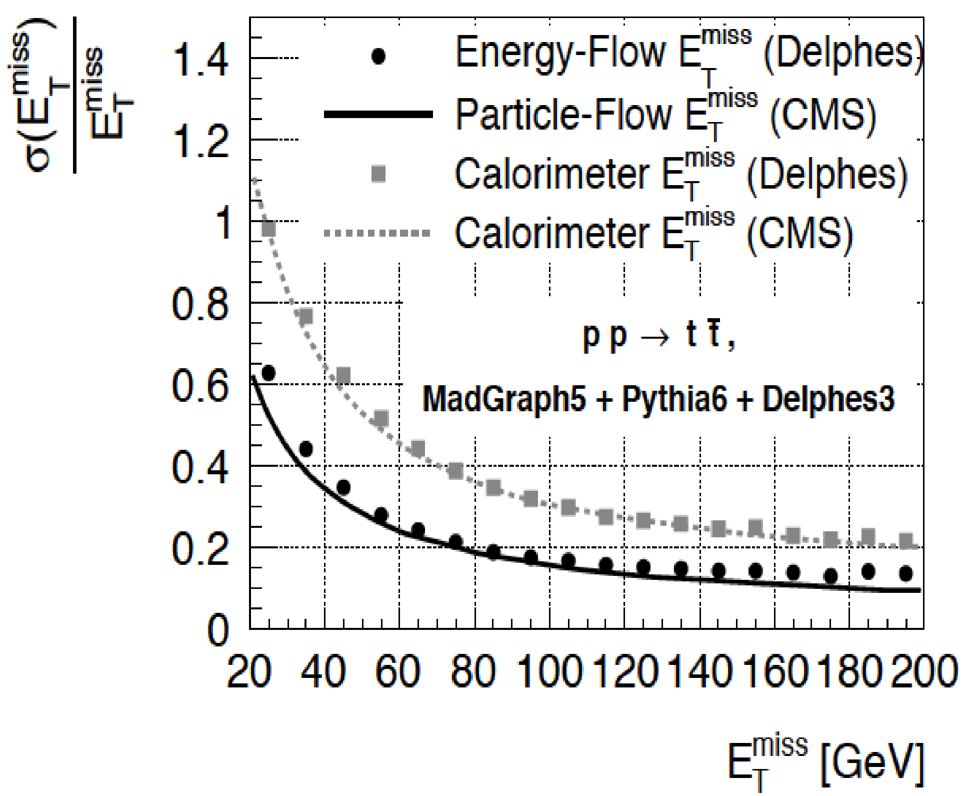
The modules: Jets / E_T^{miss} / H_T

- Delphes uses **FastJet** libraries for jet clustering
- Inputs **calorimeter towers** or "**particle-flow**" objects

```
module FastJetFinder FastJetFinder {
#  set InputArray Calorimeter/towers
  set InputArray EFlowMerger/eflow

  set OutputArray jets

  # algorithm: 1 CDFJetClu, 2 MidPoint, 3 SIScone, 4 kt, 5 Cambridge/Aachen, 6 antikt
  set JetAlgorithm 6
  set ParameterR 0.7

  set ConeRadius 0.5
  set SeedThreshold 1.0
  set ConeAreaFraction 1.0
  set AdjacencyCut 2.0
  set OverlapThreshold 0.75

  set MaxIterations 100
  set MaxPairSize 2
  set Iratch 1

  set JetPTMin 20.0
}
```

Validation: Particle-Flow





→ good agreement

Pile-Up

- **Pile-up** is implemented in Delphes **since version 3.0.4**
  - **mixes** N minimum bias events with hard event sample
  - spreads **poisson(N)** events along z-axis with configurable spread
  - rotate event by random angle φ wrt z-axis

- **Charged** Pile-up subtraction (most effective if used with PF algo)

  - if **z < |Zres|** keep all **charged and neutrals** (→ ch. particles too close to hard scattering to be rejected)

  - if **z > |Zres|** keep only **neutrals** (perfect charged subtraction)

  - allows user to tune amount of charged particle subtraction by **adjusting Z spread/resolution**

- **Residual** eta dependent pile-up substraction is needed for jets and isolation.
  - Use the FastJet Area approach (Cacciari, Salam, Soyez)
    - compute ρ = event pile-up density
    - jet correction : $pT \rightarrow pT - \rho A$ (JetPileUpSubtractor)
    - isolation : $\sum pT \rightarrow \sum pT - \rho \pi R^2$ (Isolation module itself)

UCL Université catholique de Louvain

Pile-Up

DELPHES fast simulation

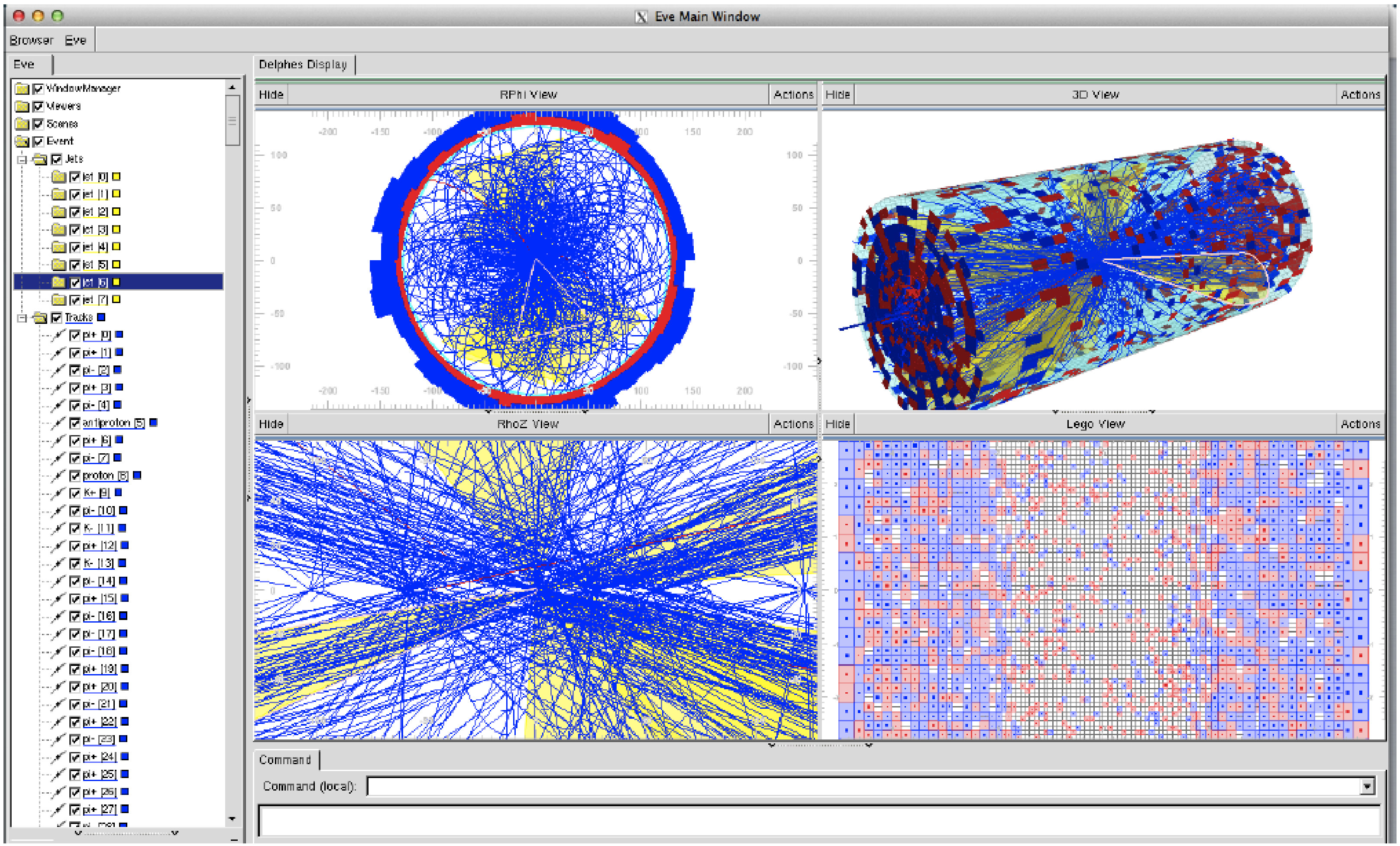
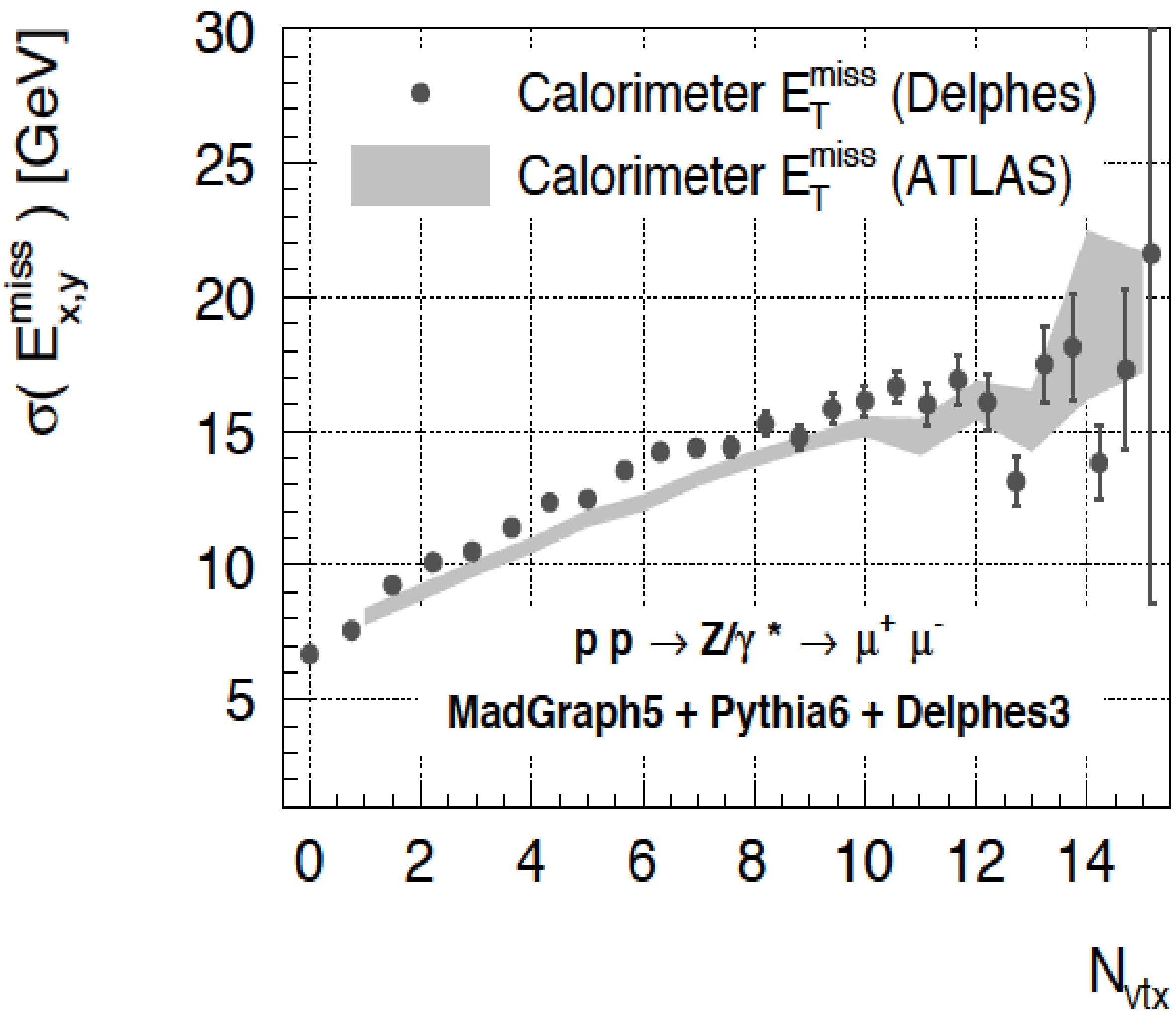


Figure 3. QCD event with 50 pile-up interactions shown with the DELPHES event display based on the ROOTEVE libraries [12]. Transverse view (top left), longitudinal view (bottom left), 3D view (top right), (η,ϕ) view (bottom right).

Validation: Pile-Up



→ **good agreement**

New Features

b-tagging

Parametrized **b-tagging**:
- Check if there is a b,c-quark in the cone of size DeltaR
- Apply a **parametrized Efficiency** (PT, eta)

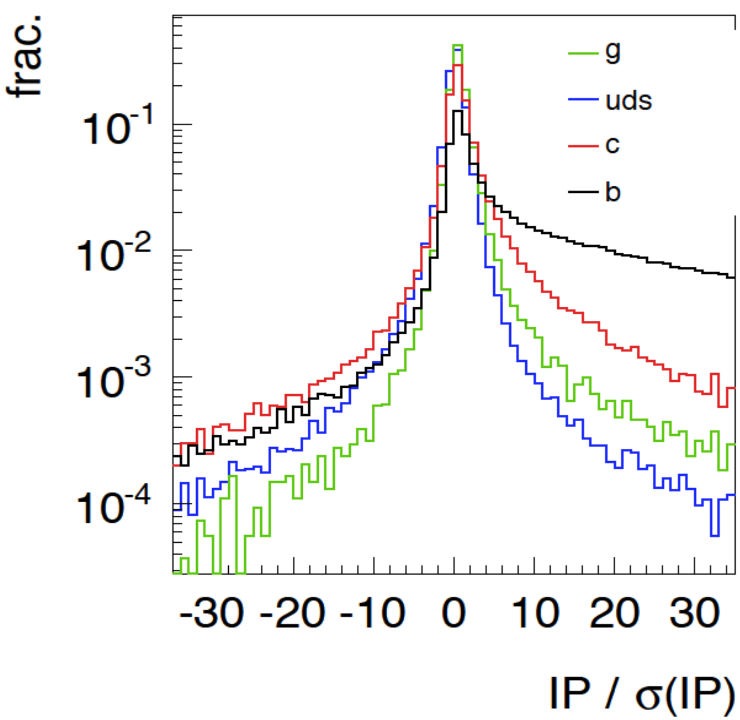
```
module BTagging BTagging {
  set PartonInputArray Delphes/partons
  set JetInputArray JetEnergyScale/jets

  set BitNumber 0
  set DeltaR 0.5
  set PartonPTMin 1.0
  set PartonEtaMax 2.5

  # default efficiency formula (misidentification rate)
  add EfficiencyFormula {0} {0.001}

  # efficiency formula for c-jets (misidentification rate)
  add EfficiencyFormula {4} {                                      (pt <= 15.0) * (0.000) + \
                                                (abs(eta) <= 1.2) * (pt > 15.0) * (0.2*tanh(pt*0.03 - 0.4)) + \
                              (abs(eta) > 1.2 && abs(eta) <= 2.5) * (pt > 15.0) * (0.1*tanh(pt*0.03 - 0.4)) + \
                              (abs(eta) > 2.5)                                  * (0.000)}

  # efficiency formula for b-jets
  add EfficiencyFormula {5} {                                      (pt <= 15.0) * (0.000) + \
                                                (abs(eta) <= 1.2) * (pt > 15.0) * (0.5*tanh(pt*0.03 - 0.4)) + \
                              (abs(eta) > 1.2 && abs(eta) <= 2.5) * (pt > 15.0) * (0.4*tanh(pt*0.03 - 0.4)) + \
                              (abs(eta) > 2.5)                                  * (0.000)}
}
```

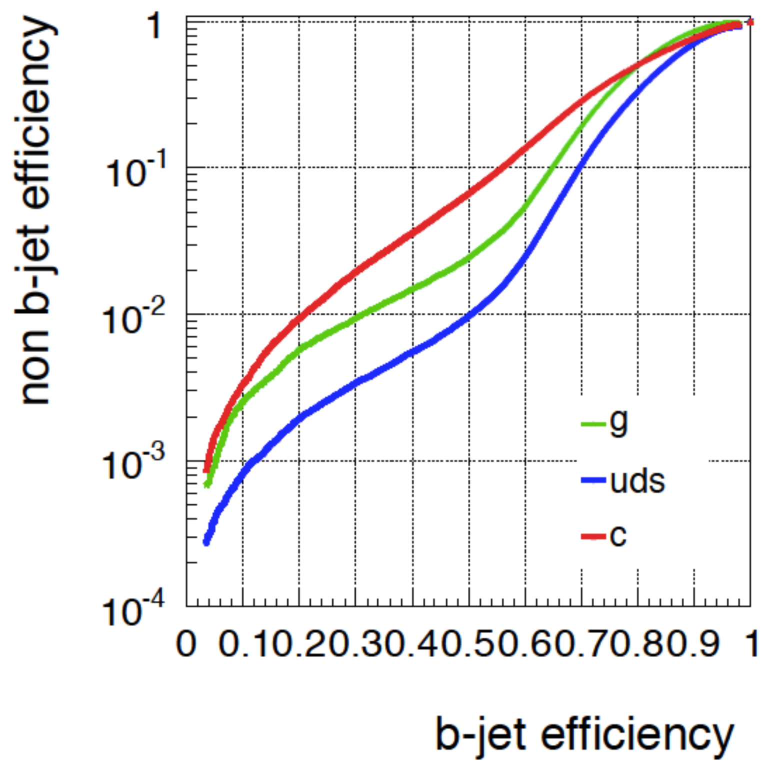
→ perfectly reproduces existing performances

→ not predictive

Track counting b-tagging

- Track parameters (p_T, d_{XY}, d_Z) derived from **track fitting** in real experiments
- In Delphes we can **smear** directly $\mathbf{d_{XY}}$**,** $\mathbf{d_Z}$ according to **(**$\mathbf{p_T}$**,** $\boldsymbol{\eta}$**)** of the track
- **Count tracks** within jet with **large impact parameter** significance.





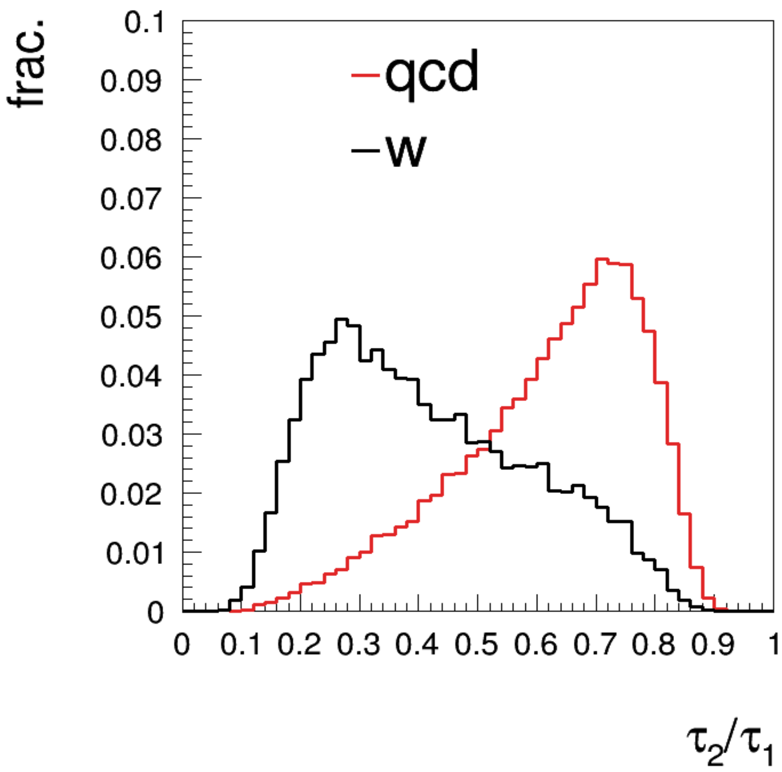
→ although very simple is predictive

→ ignore correlations among track parameters

N-subjettiness and N-jettiness

JHEP 1103:015 (2011), JHEP 1202:093 (2012) and JHEP 1404:017 (2014)

- very useful for identifying **sub-structure** of highly-boosted jets.
- build ratios τ_N / τ_M to **discriminate** between **N or M-prong**



- Embedded in FastJetFinder module
- Variables τ_1, τ_2, .. , τ_5 saved as jet members (N-subjettiness)

```
###################
# N-subjettiness
##################

module FastJetFinder FastJetFinder {
#  set InputArray Calorimeter/towers
  set InputArray EFlowMerger/eflow

  set OutputArray jets

  # algorithm: 1 CDFJetClu, 2 MidPoint, 3 SIScone, 4 kt,
  #            5 Cambridge/Aachen, 6 antikt, 7 antikt wta, 8 Njettiness

  set JetAlgorithm 7
  set ParameterR 0.5

  set ComputeNsubjettiness 1
  set Beta 1.0

  #axis mode: 1 wta, 2 wta optimized, 3 kt, 4 kt optimized
  set AxisMode 1

  set JetPTMin 1000.0
}
```

Thanks to A. Larkowski for help

Delphes and hh@100TeV

Delphes and hh@100TeV

- Delphes has been designed to deal with **high number of hadrons** environment**:**
  - Jets, MET and object isolation are modeled realistically
  - pile-up subtraction (FastJet Area method, Charged Hadron Subtraction)
  - pile-up JetId
- Recent improvements (Delphes 3.1.0)
  - **different segmentation** for ECAL and HCAL
  - Impact parameter smearing: allow for **predictive b-tagging** (now parametrized)
  - **jet substructure** and for boosted objects (N-(sub)jettiness)
  - Included dummy configuration card for future collider studies (use with caution!)

Delphes and hh@100TeV

Delphes can be used **right-away** for **hh@100TeV** studies ...

What can you do with Delphes?

- reverse engineering

  → you have some target for jet invariant mass resolution
  what granularity and resolution are needed to achieve it?

- impact of pile-up on isolation, jet structure, multiplicities ...

In which context?

- **preliminary physics studies** can be performed in **short time** (e.g SnowMass)

- can be used **in parallel** with full detector simulation

- flexible software structure allows **integration** in other frameworks (can be called from others programs, see manual)

Conclusions

- **Delphes 3** has been out for one year now, with **major improvements**:

  - modularity
  - pile-up
  - visualization tool based on ROOT EVE
  - default cards giving results on par with published performance from LHC experiments
  - fully integrated within MadGraph5

- Delphes 3.1 can be used right away for fast and realistic simulation of h-h collisions

- Continuous development (IP b-tagging, Nsubjettiness, Calorimeters ...)

- Delphes TUTORIAL on May 8th in CERN

Website and manual:

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

People

Jerome de Favereau
Christophe Delaere
Pavel Demin
Andrea Giammanco
Vincent Lemaitre
Alexandre Mertens
Michele Selvaggi

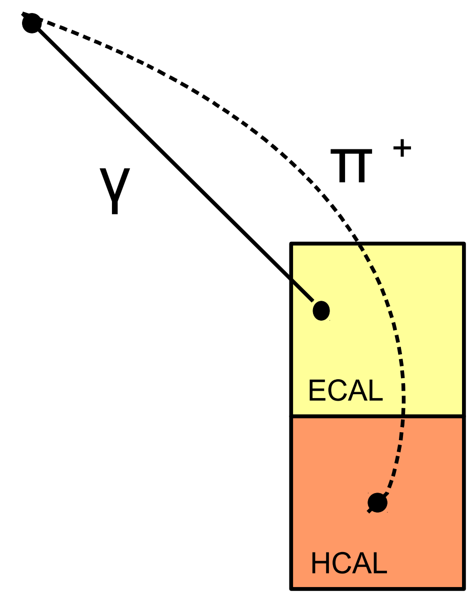
the community ...

Back-up

The modules: Particle-Flow Emulation

Example 2: A pion (10 GeV) and a photon (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 $E^{PF\text{-}tower} = 4 + 18$ GeV

Separate neutral and charged calo deposits has crucial implications for pile-up subtraction

No separation between “Photons” and “Neutral Hadrons” in the output.

The modules: Leptons and photons reconstruction

- Muons/electrons
  - **identified** via their PDG id
  - muons do not deposit energy in calo (independent smearing parameterized in p_T and η)
  - electrons smeared according to tracker and ECAL resolution

- Isolation:

$$I(P) = \frac{\sum_{i \neq P}^{\Delta R < R,\ p_T(i) > p_T^{min}} p_T(i)}{p_T(P)}$$

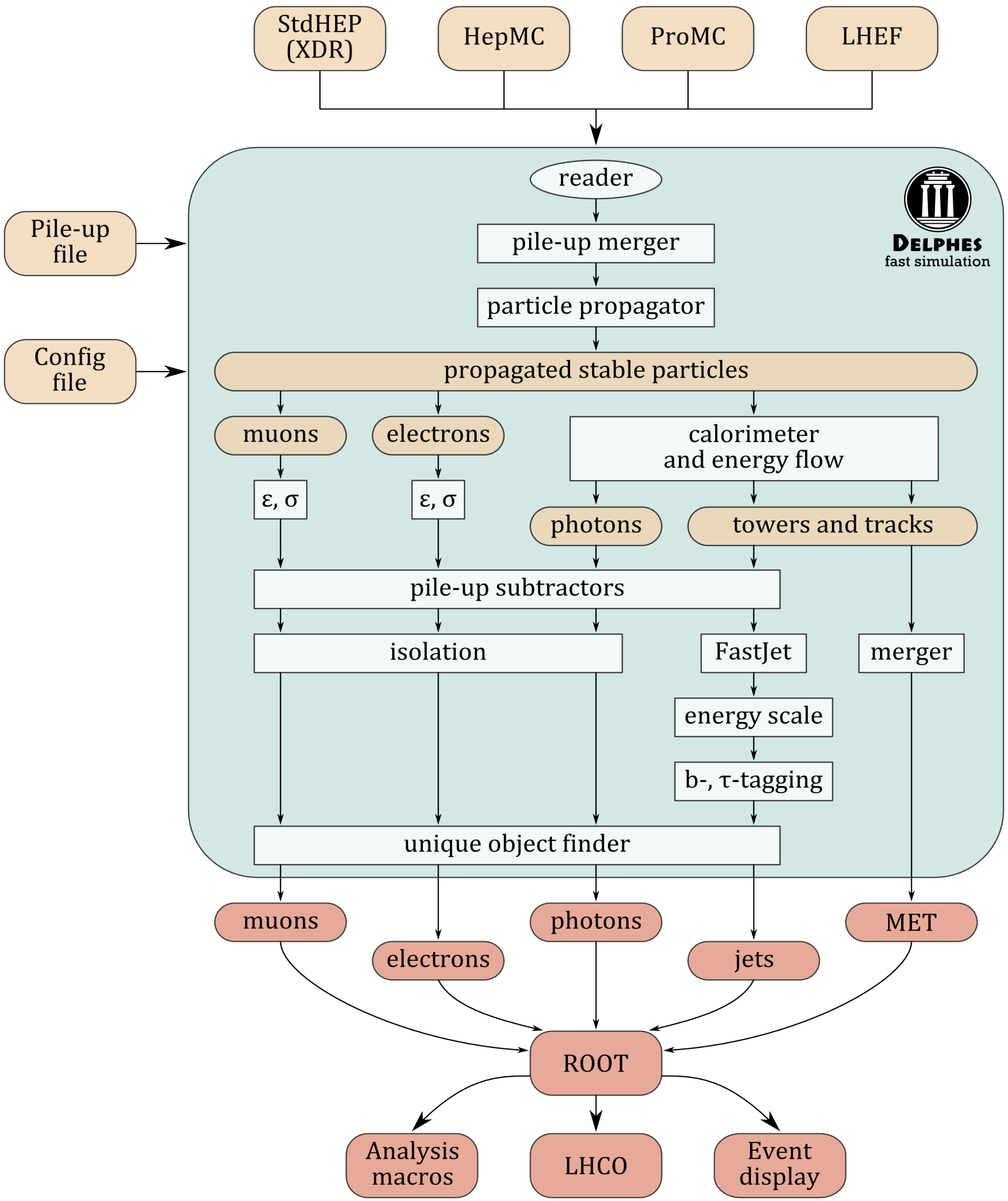
If **I(P) < Imin**, the lepton is **isolated**

User can specify parameters I_{min}, ΔR, p_T^{min}

```
##################
# Photon isolation
##################

module Isolation PhotonIsolation {
  set CandidateInputArray PhotonEfficiency/photons
  set IsolationInputArray EFlowMerger/eflow

  set OutputArray photons

  set DeltaRMax 0.5
  set PTMin 0.5
  set PTRatioMax 0.1
}
```

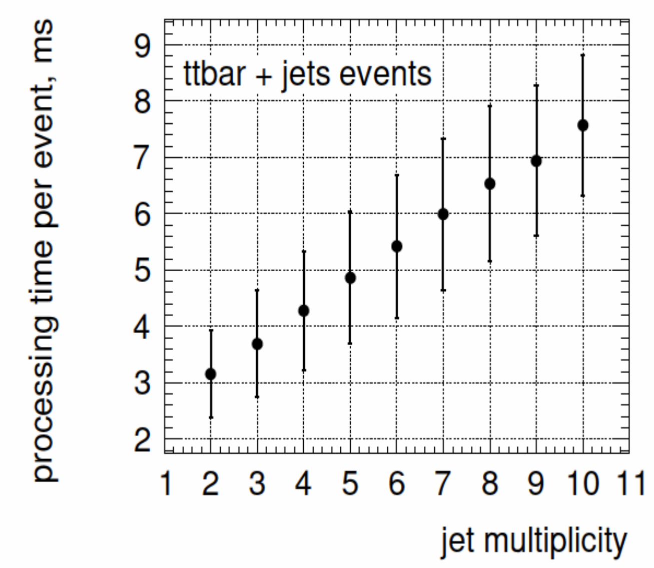
No fakes, punch-through, brehmstrahlung, conversions

The Delphes project: A modular structure



- Every Object in Delphes is a **Candidate**.
- All **modules** consume and produce **Arrays** of **Candidates**.
- Any module can **access** Arrays produced by other modules using **ImportArray** method: *ImportArray("ModuleName/arrayName")*
- The Delphes team provides a set of modules.
- A user can create **new modules** and define its **own sequence**.

The Delphes Project: CPU time

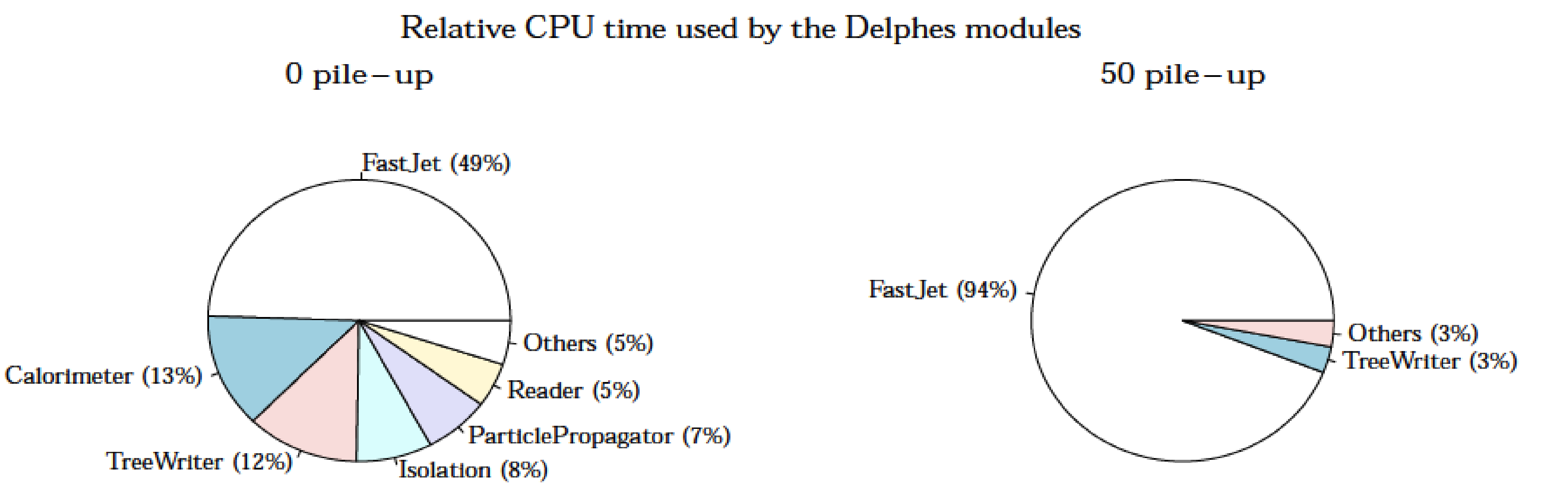


Delphes **reconstruction time** per event:

0 Pile-Up = 1 ms

150 Pile-Up = 1 s

Mainly spent in the FastJet algorithm:



The Delphes Project: disk space

Disk **space** for 10k ttbar events (upper limit, store all constituents):

0 Pile-Up = 300 Mb

100 Pile-Up = 3 Gb

Mainly taken by list of MC particles and Calo towers: