'Track-Based' Particle Flow

Outline:

- Particle Flow Algorithms available for Marlin
- 'Cluster-Based' vs. 'Track-Based' PFlow approach
- 'Track-Based' Particle Flow Algorithm in Marlin
- MIP Stub finding
- Summary and Outlook

Particle Flow Algorithms available for Marlin

Wolf (A. Raspereza) and **PandoraPFA** (M. Thomson):

- Wolf: $\Delta E/E \approx \mathbf{0.5}/\sqrt{E}$ ($RMS_{90\%}$) for $Z^0 \rightarrow$ uds @ 91.2 GeV
- PandoraPFA: $\Delta E/E \approx \mathbf{0.3}/\sqrt{E}$ ($RMS_{90\%}$) for $Z^0 \rightarrow$ uds @ 91.2 GeV

✔ both perform **'reasonable'** @ 91.2 GeV, benchmark: PandoraPFA

✗ performance of both **degrade rapidly** with increasing jet energy

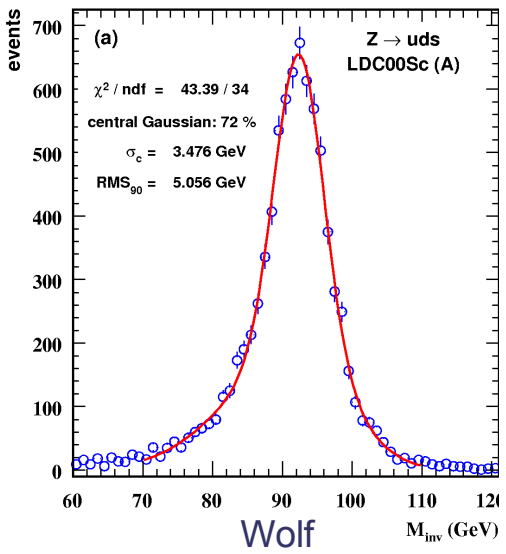
➢ both are **'cluster-based'** algorithms (PandoraPFA: tracks ↔ cluster-ass.)

'track-based' algorithm should perform better

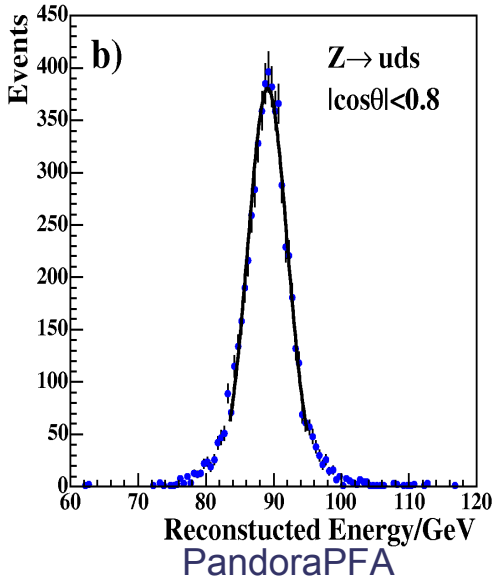
➔ might be **more complex**

modular approach

➔ provide a 'tool-box' for **various** Particle Flow Algorithms

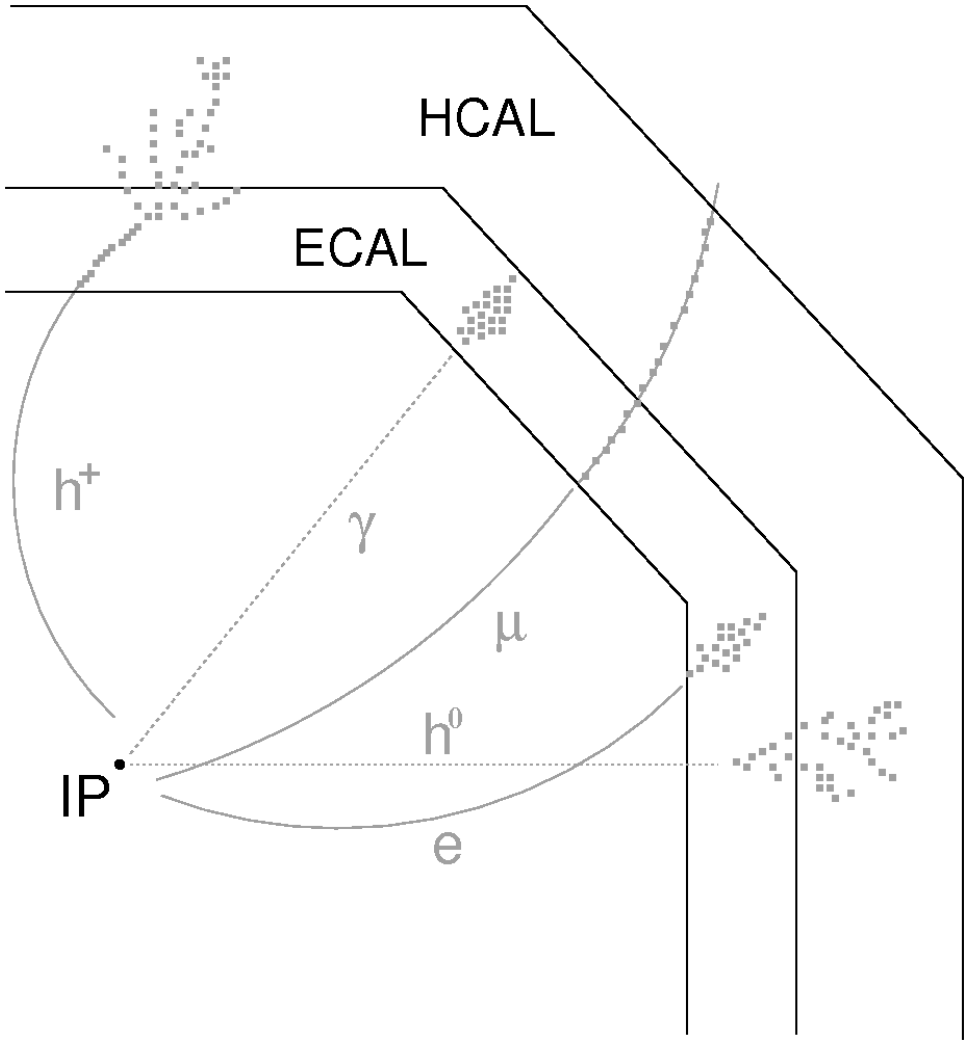


Wolf



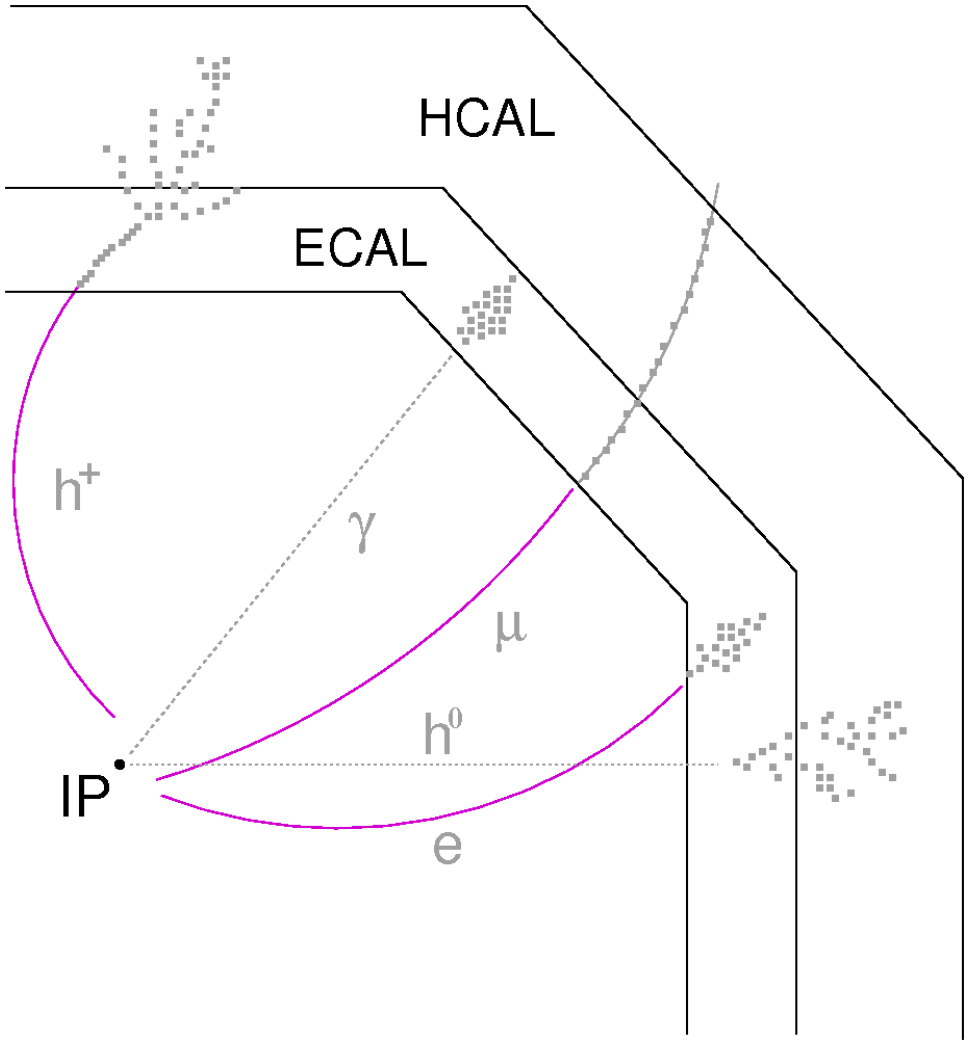
PandoraPFA

Particle Flow: 'Cluster-Based' Approach



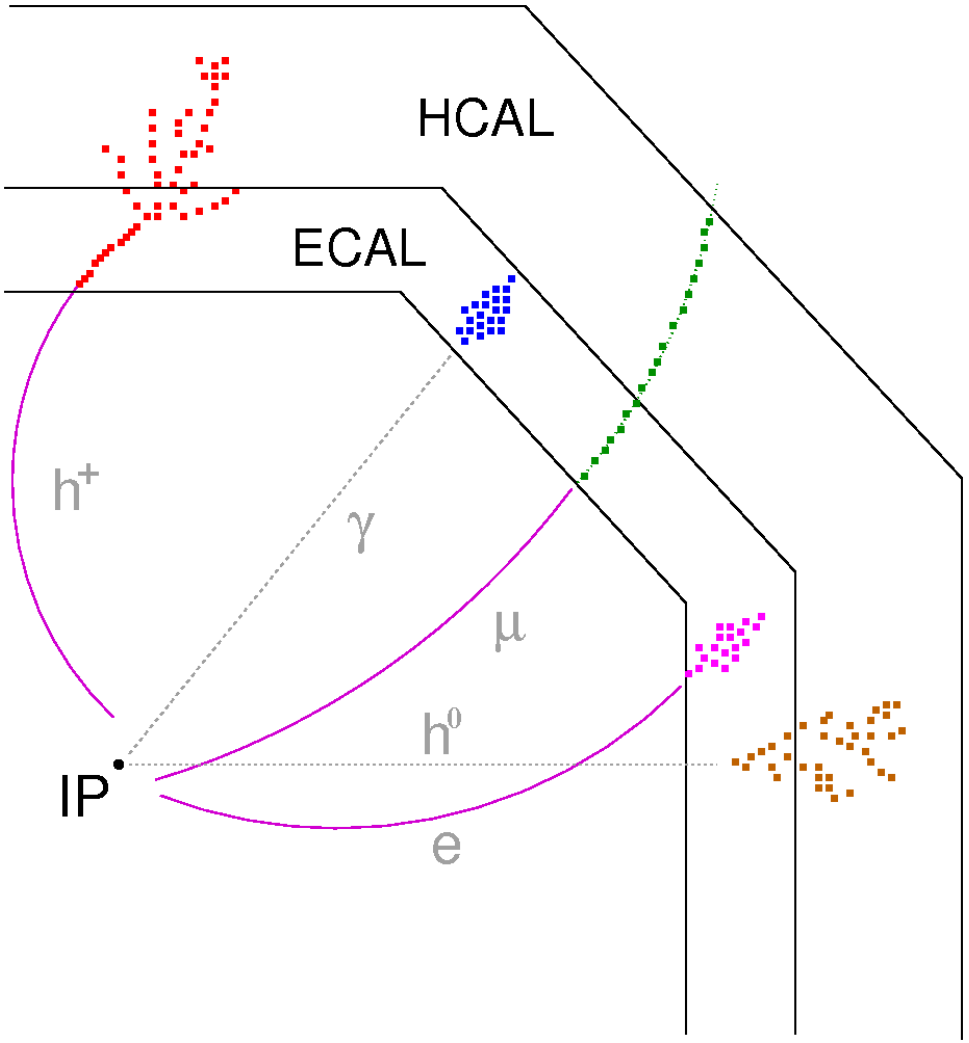
Particle Flow: 'Cluster-Based' Approach

1. tracking (VTX, SIT, TPC...)



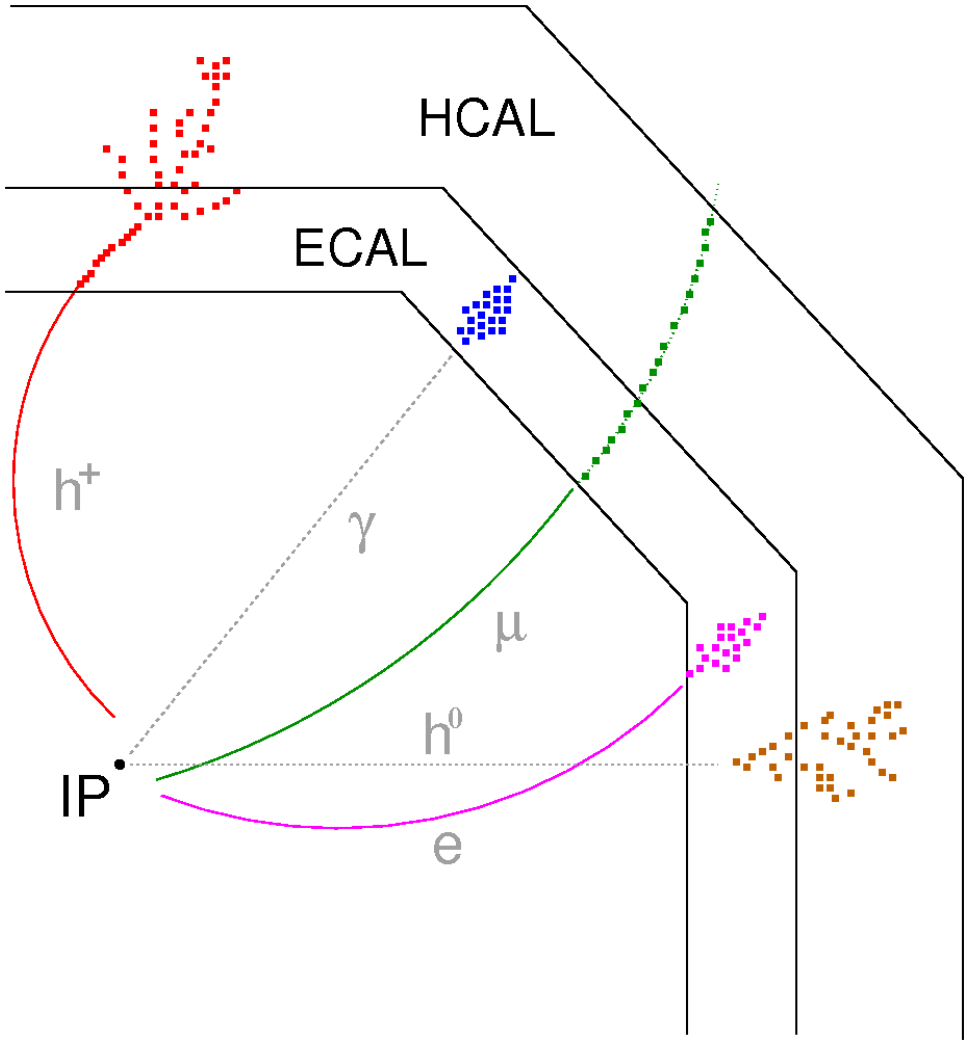
Particle Flow: 'Cluster-Based' Approach

1. tracking (VTX, SIT, TPC...)
2. clustering (ECAL and HCAL)
   - → independent
   - → different algorithms



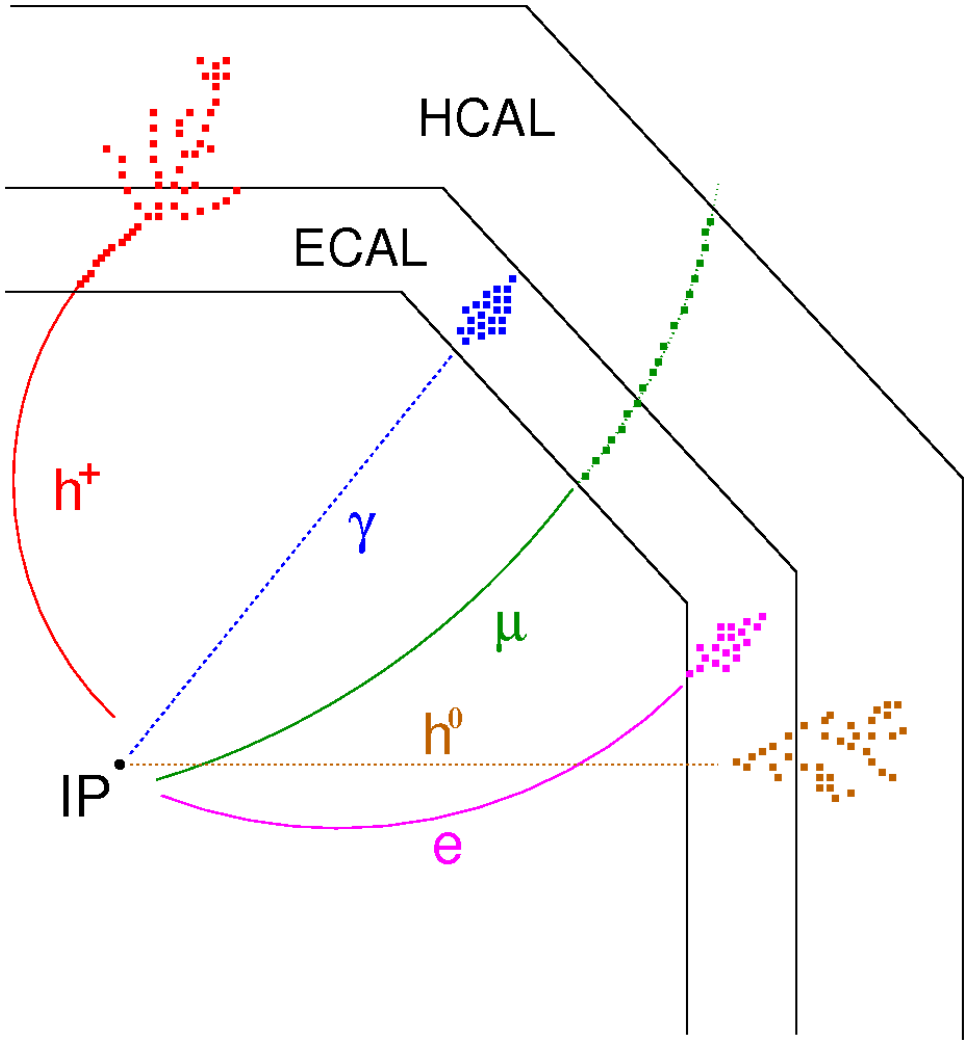
Particle Flow: 'Cluster-Based' Approach

1. tracking (VTX, SIT, TPC...)
2. clustering (ECAL and HCAL)
   - → independent
   - → different algorithms
3. track cluster matching
   - → proximity criteria

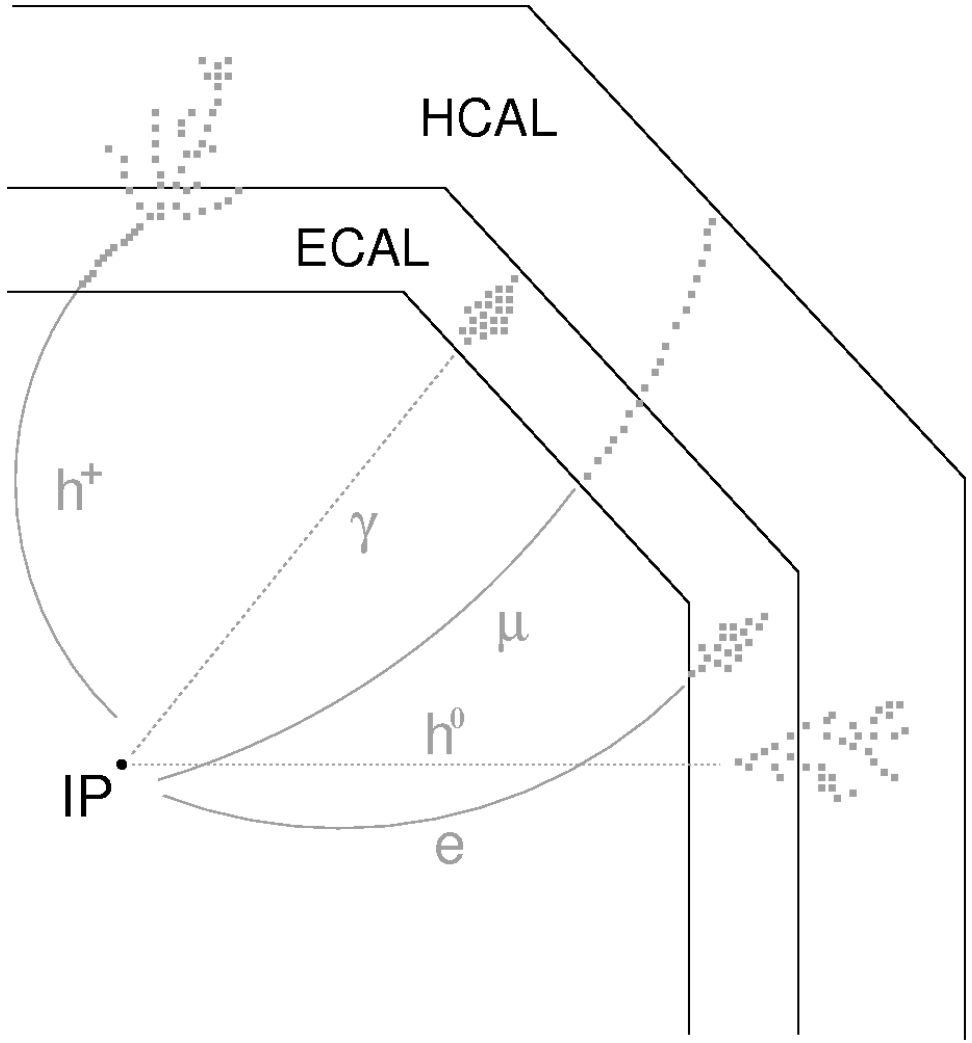


Particle Flow: 'Cluster-Based' Approach

1. tracking (VTX, SIT, TPC...)
2. clustering (ECAL and HCAL)
   - → independent
   - → different algorithms
3. track cluster matching
   - → proximity criteria
4. particle ID
   - → e.g. fraction of energy in ECAL/HCAL
   - → $e^{+/-}$, $\mu^{+/-}$, $h^{+/-}$
   - → assign clusters w/o tracks to neutral objects (γ, h^0)

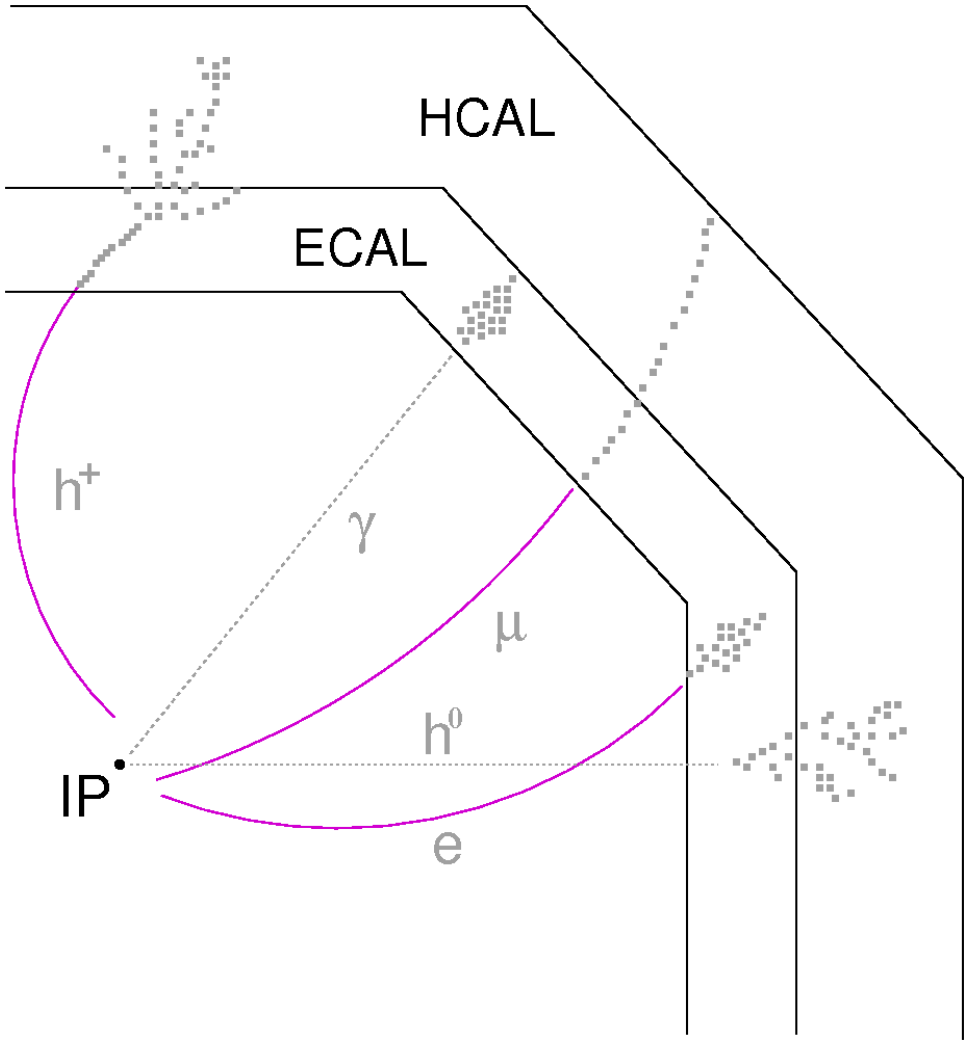


Particle Flow: 'Track-Based' Approach



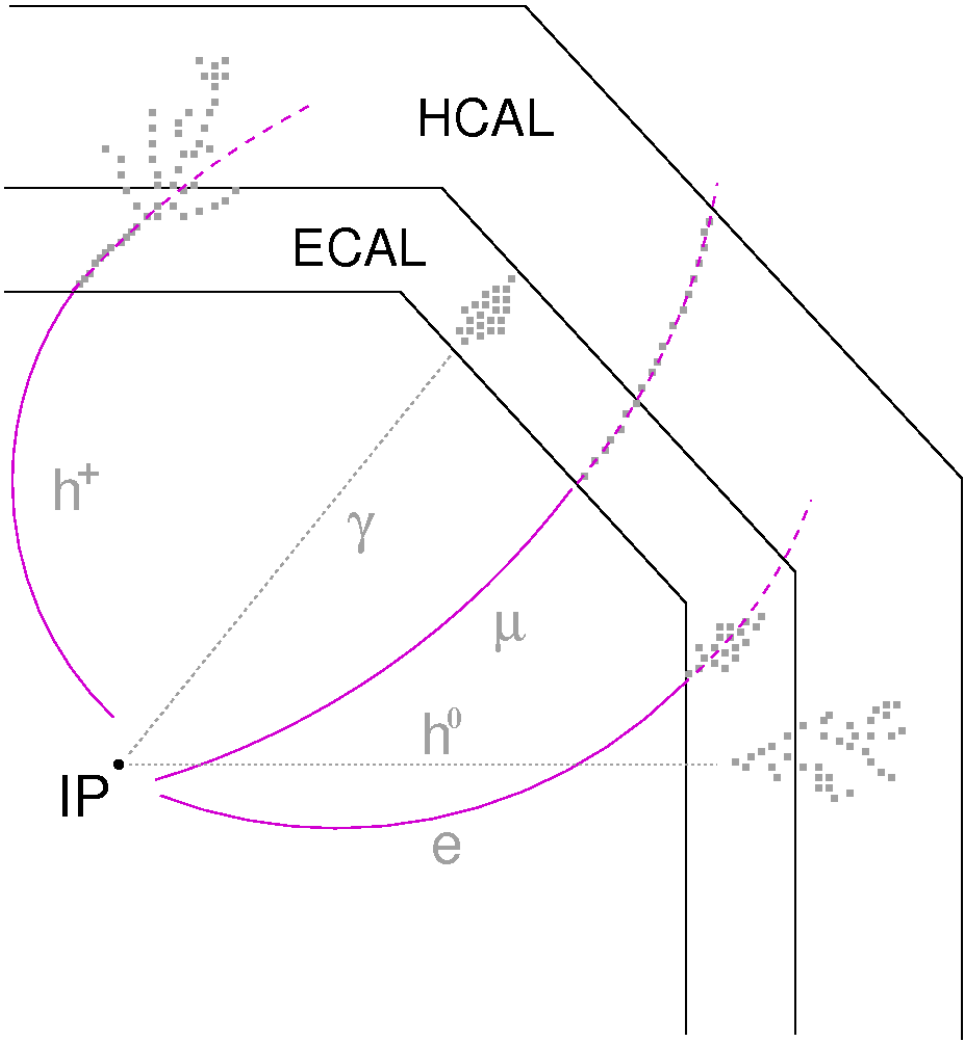
Particle Flow: 'Track-Based' Approach

1. tracking (VTX, SIT, TPC...)



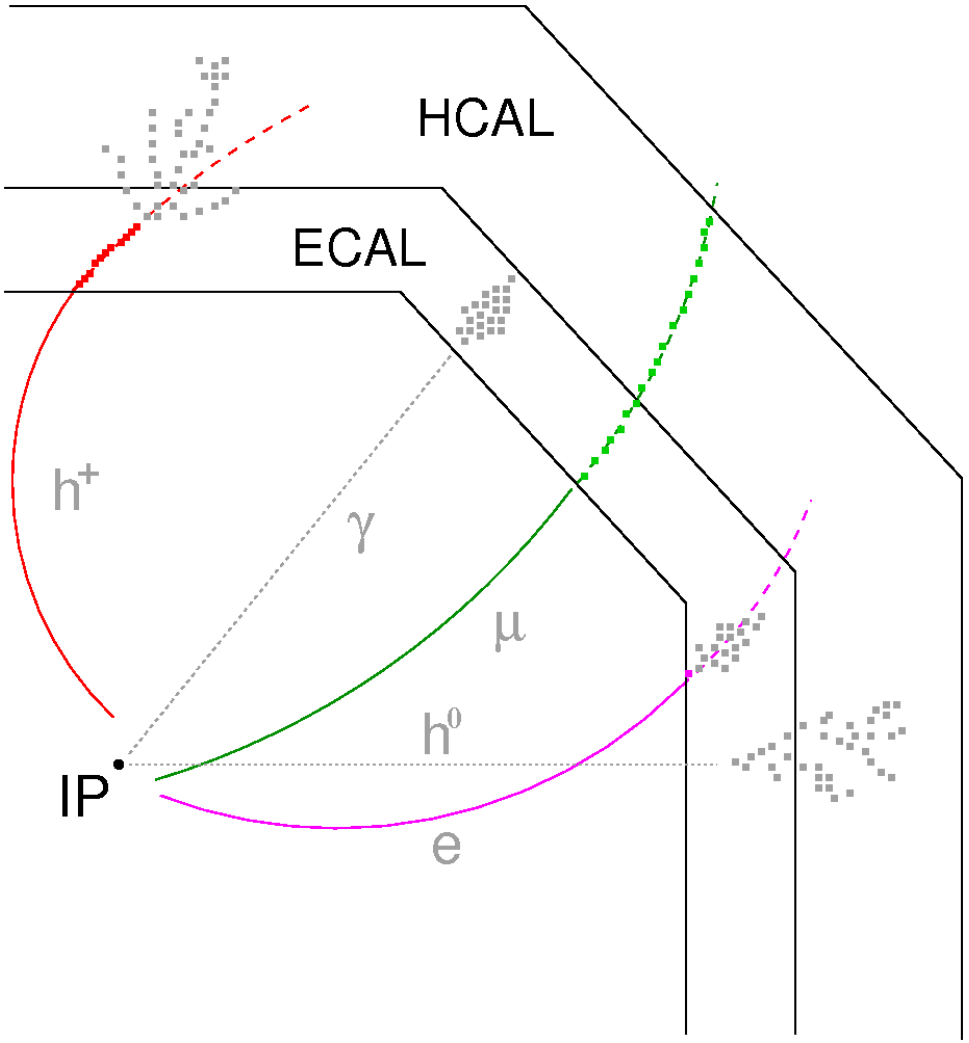
Particle Flow: 'Track-Based' Approach

1. tracking (VTX, SIT, TPC...)
2. extrapolate tracks into Calorimeter



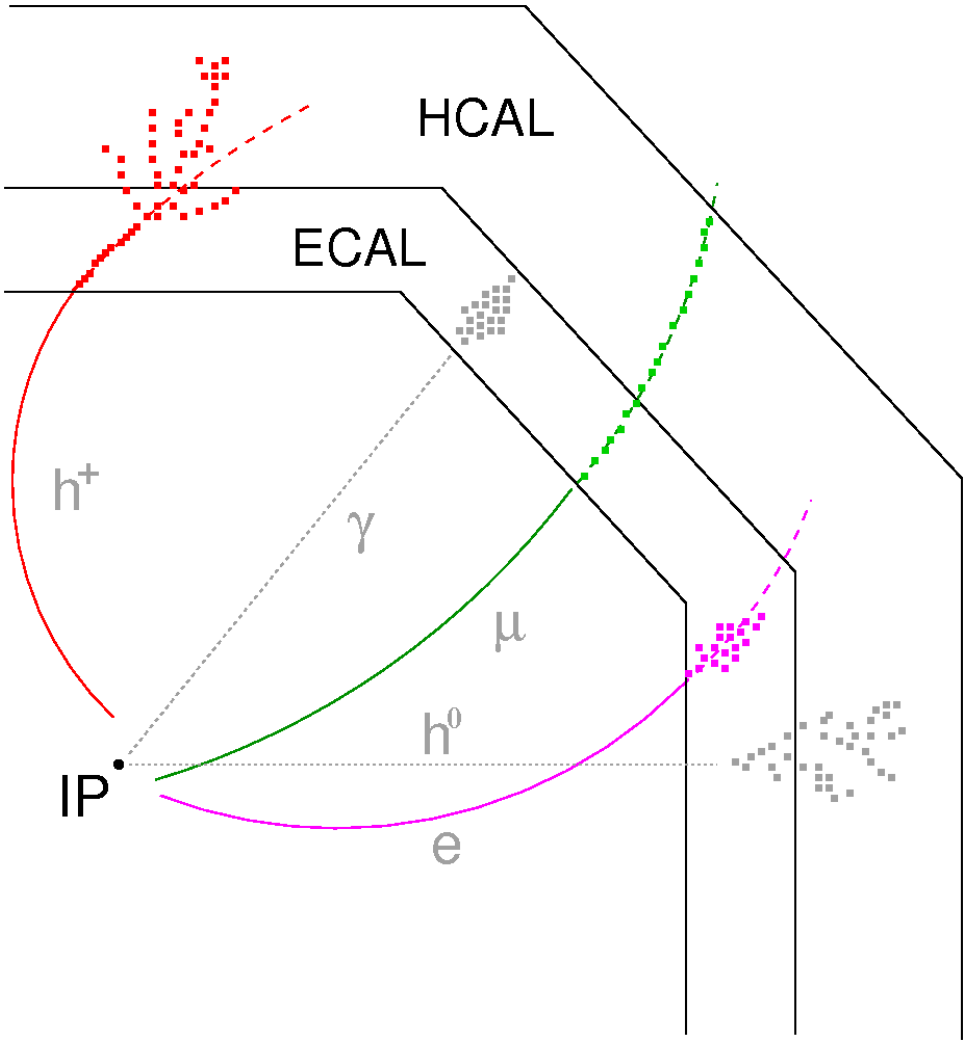
Particle Flow: 'Track-Based' Approach

1. tracking (VTX, SIT, TPC...)
2. extrapolate tracks into Calorimeter
3. assign MIP stub to track
   - ➔ get $\mu^{+/-}$ as well



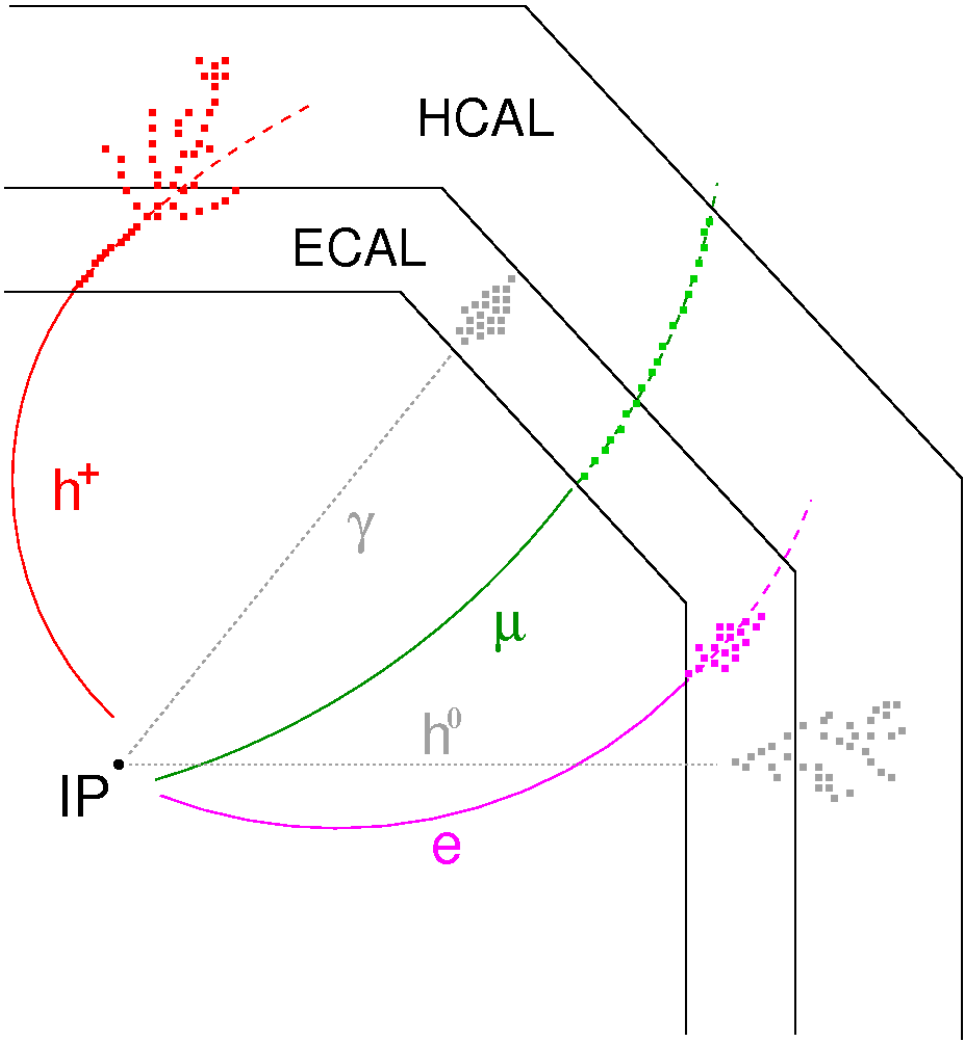
Particle Flow: 'Track-Based' Approach

1. tracking (VTX, SIT, TPC...)
2. extrapolate tracks into Calorimeter
3. assign MIP stub to track
   - → get $\mu^{+/-}$ as well
4. clustering (ECAL and HCAL)
   - → variable, depending on track
   - → different algorithms



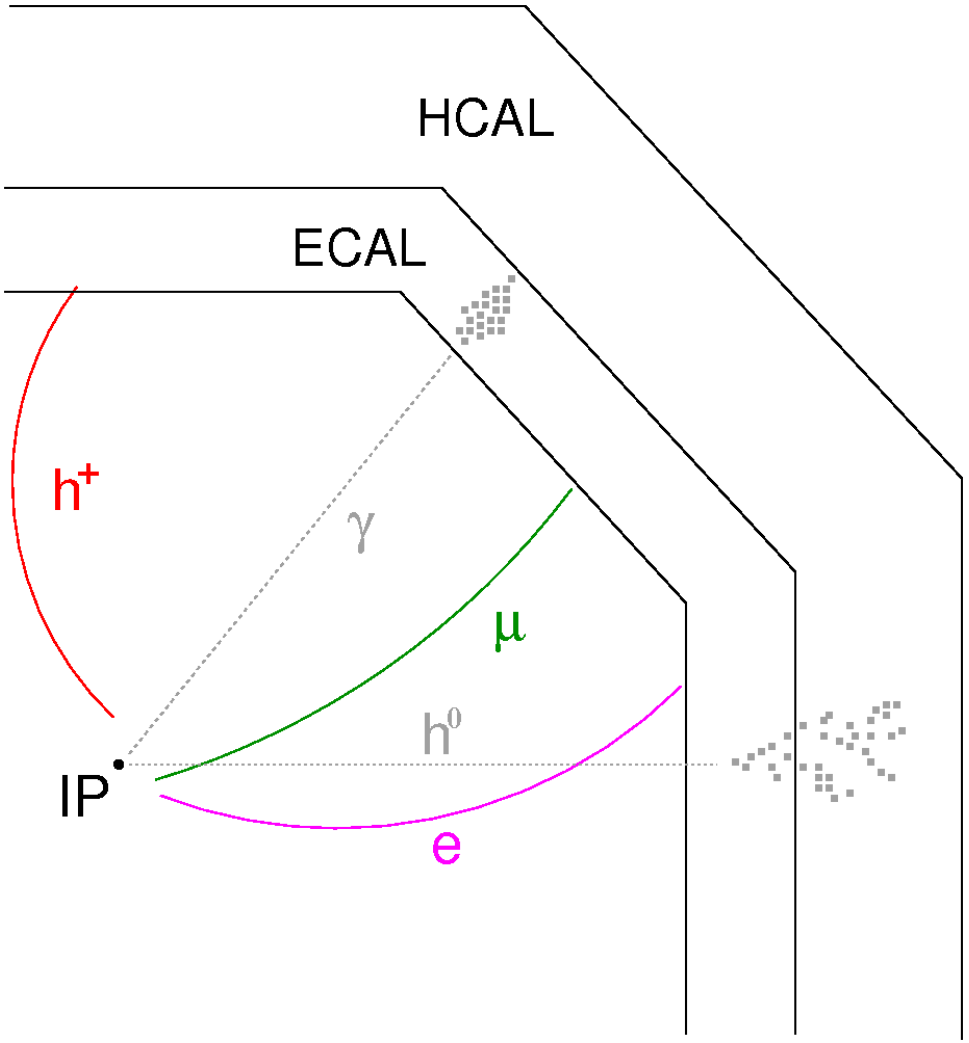
Particle Flow: 'Track-Based' Approach

1. tracking (VTX, SIT, TPC...)
2. extrapolate tracks into Calorimeter
3. assign MIP stub to track
   - → get $\mu^{+/-}$ as well
4. clustering (ECAL and HCAL)
   - → variable, depending on track
   - → different algorithms
5. particle ID for $e^{+/-}$, $\mu^{+/-}$, $h^{+/-}$
   - → e.g. fraction of energy in ECAL/HCAL



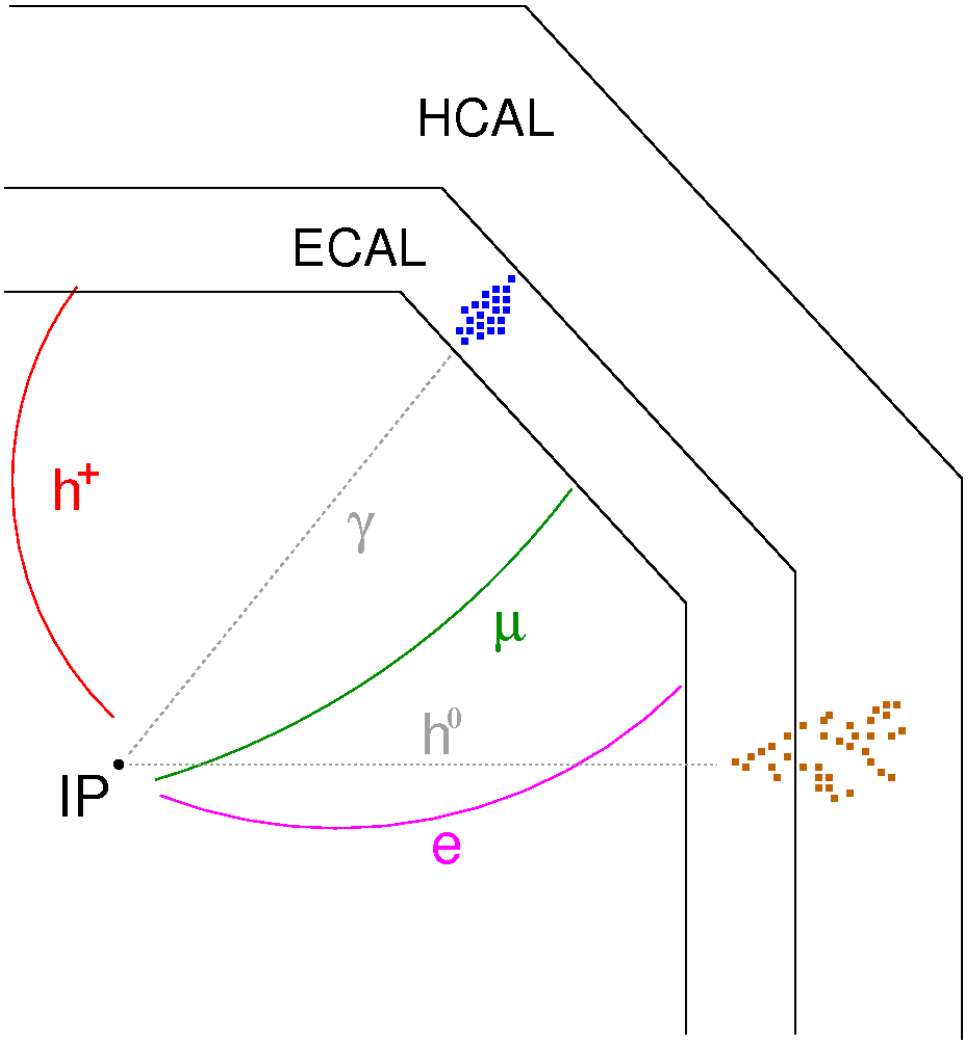
Particle Flow: 'Track-Based' Approach

1. tracking (VTX, SIT, TPC...)
2. extrapolate tracks into Calorimeter
3. assign MIP stub to track
   - → get $\mu^{+/-}$ as well
4. clustering (ECAL and HCAL)
   - → variable, depending on track
   - → different algorithms
5. particle ID for $e^{+/-}$, $\mu^{+/-}$, $h^{+/-}$
   - → e.g. fraction of energy in ECAL/HCAL
6. remove 'charged' Calorimeter hits



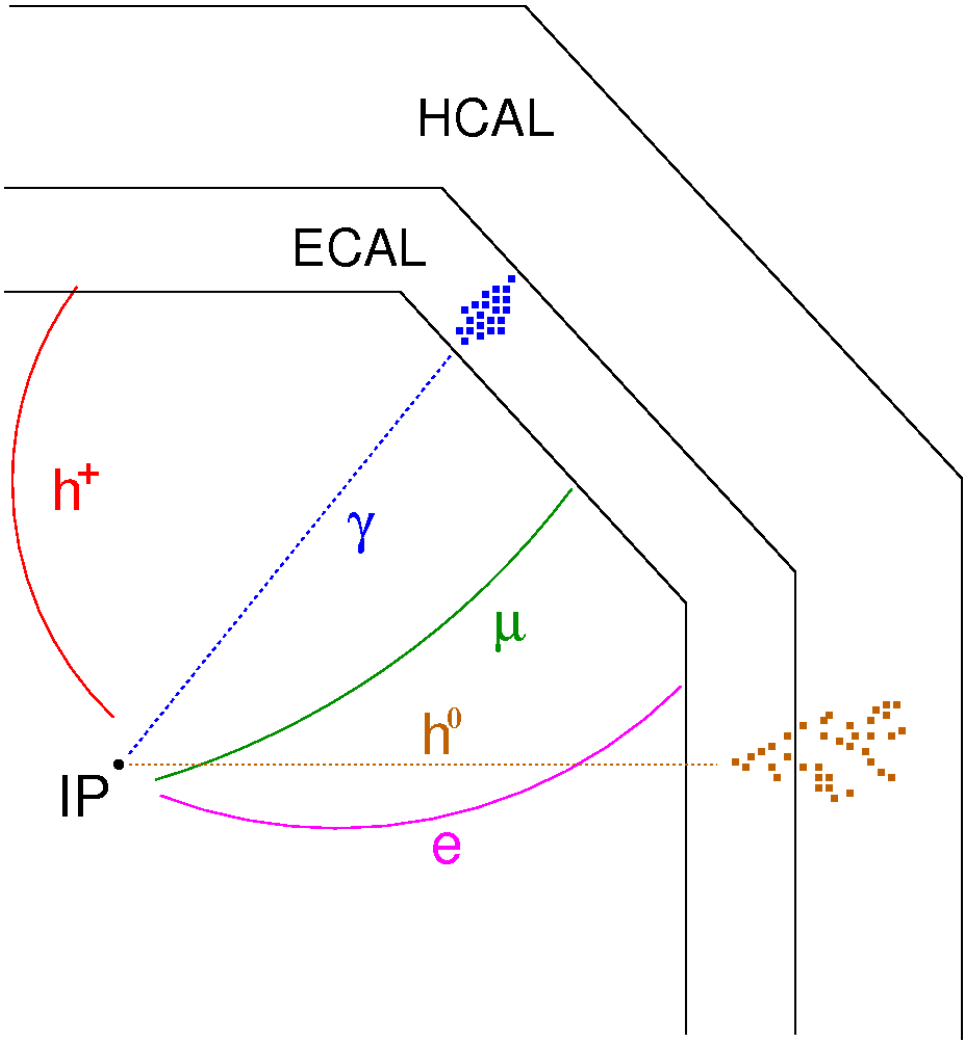
Particle Flow: 'Track-Based' Approach

1. tracking (VTX, SIT, TPC...)
2. extrapolate tracks into Calorimeter
3. assign MIP stub to track
   - → get $\mu^{+/-}$ as well
4. clustering (ECAL and HCAL)
   - → variable, depending on track
   - → different algorithms
5. particle ID for $e^{+/-}$, $\mu^{+/-}$, $h^{+/-}$
   - → e.g. fraction of energy in ECAL/HCAL
6. remove 'charged' Calorimeter hits
7. clustering on 'neutral' hits



Particle Flow: 'Track-Based' Approach

1. tracking (VTX, SIT, TPC...)
2. extrapolate tracks into Calorimeter
3. assign MIP stub to track
   - → get $\mu^{+/-}$ as well
4. clustering (ECAL and HCAL)
   - → variable, depending on track
   - → different algorithms
5. particle ID for $e^{+/-}$, $\mu^{+/-}$, $h^{+/-}$
   - → e.g. fraction of energy in ECAL/HCAL
6. remove 'charged' Calorimeter hits
7. clustering on 'neutral' hits
8. particle ID for γ, h^0



'Cluster-Based' PFlow ↔ 'Track-Based' PFlow

'Cluster-Based' Approach

- treat 'charged' and 'neutral' clusters equally
- more 'simple' (less modules)
  - → 'easy' to study
  - → faster

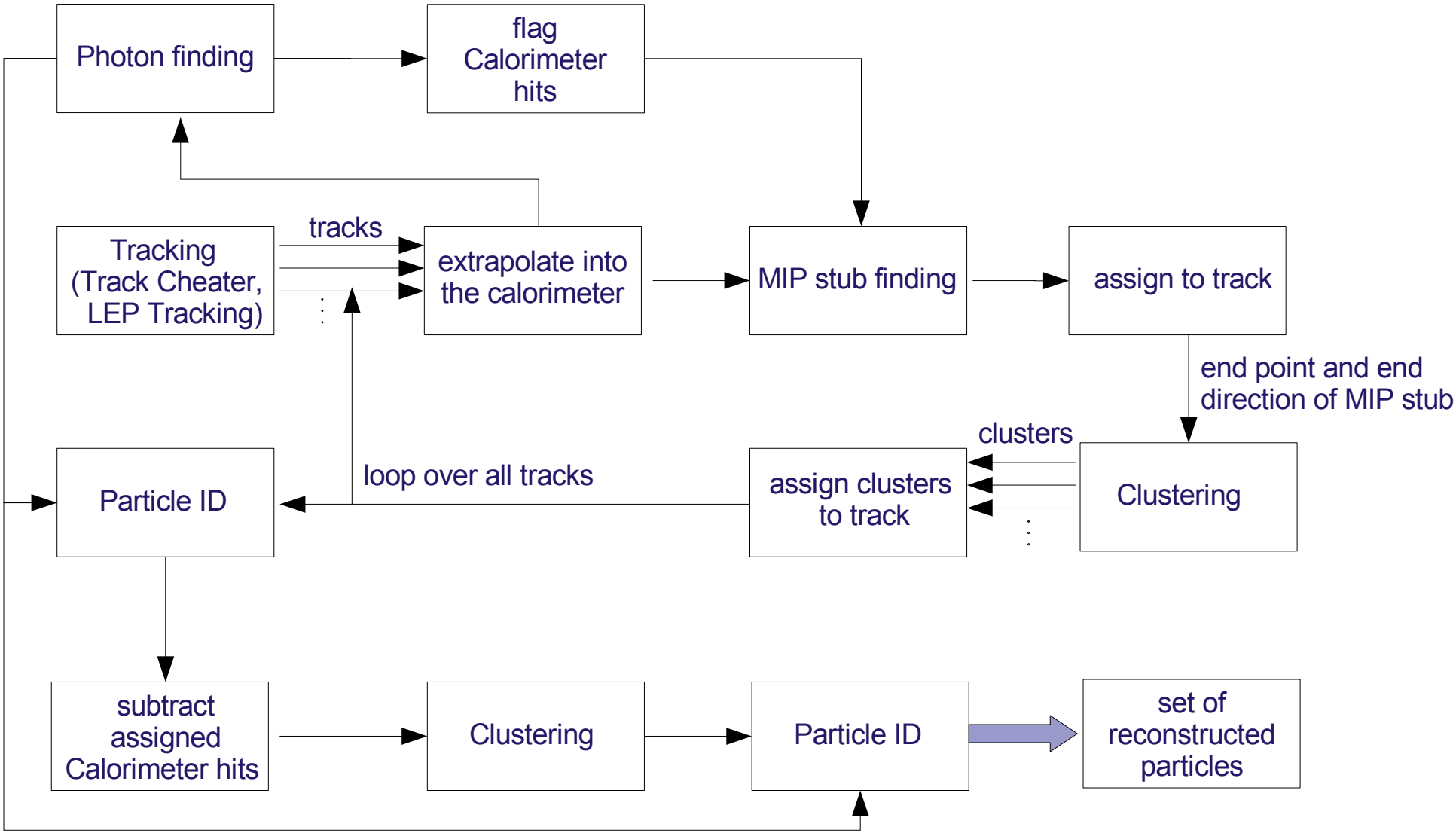
'Track-Based' Approach

- 'prefer charged' clusters
  - → potential bias
- more potential for a 'better' (Jet)Energy resolution
  - → full track information (dE/dx, ...)
  - → 'guided' clustering
- more complex
  - → systematic effects, slower

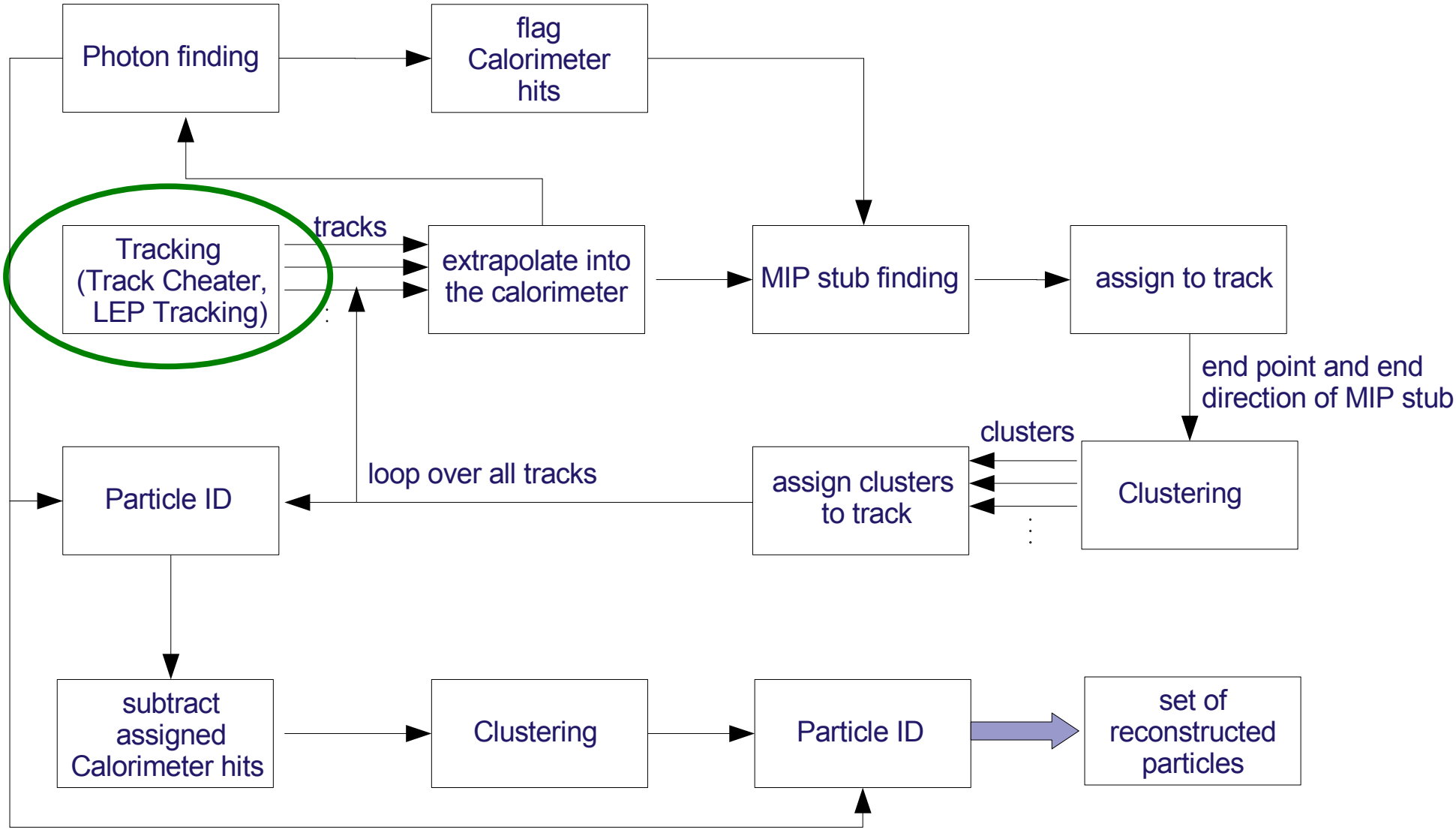
Modularity:

- → optimise / exchange modules
- → compose new PFlow algorithms
- → distribute work

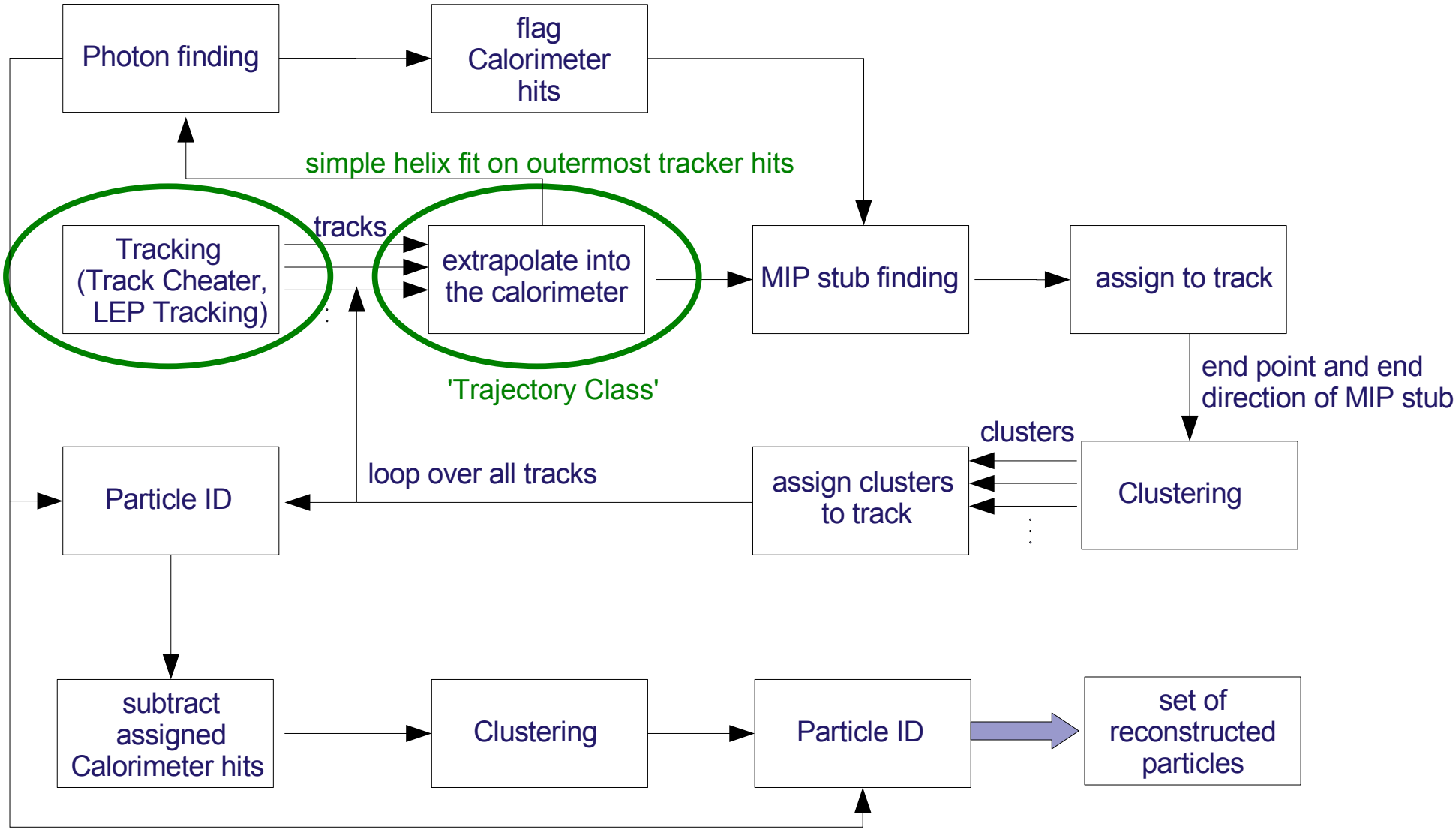
'Track-Based PFlow' in Marlin / Software Chain



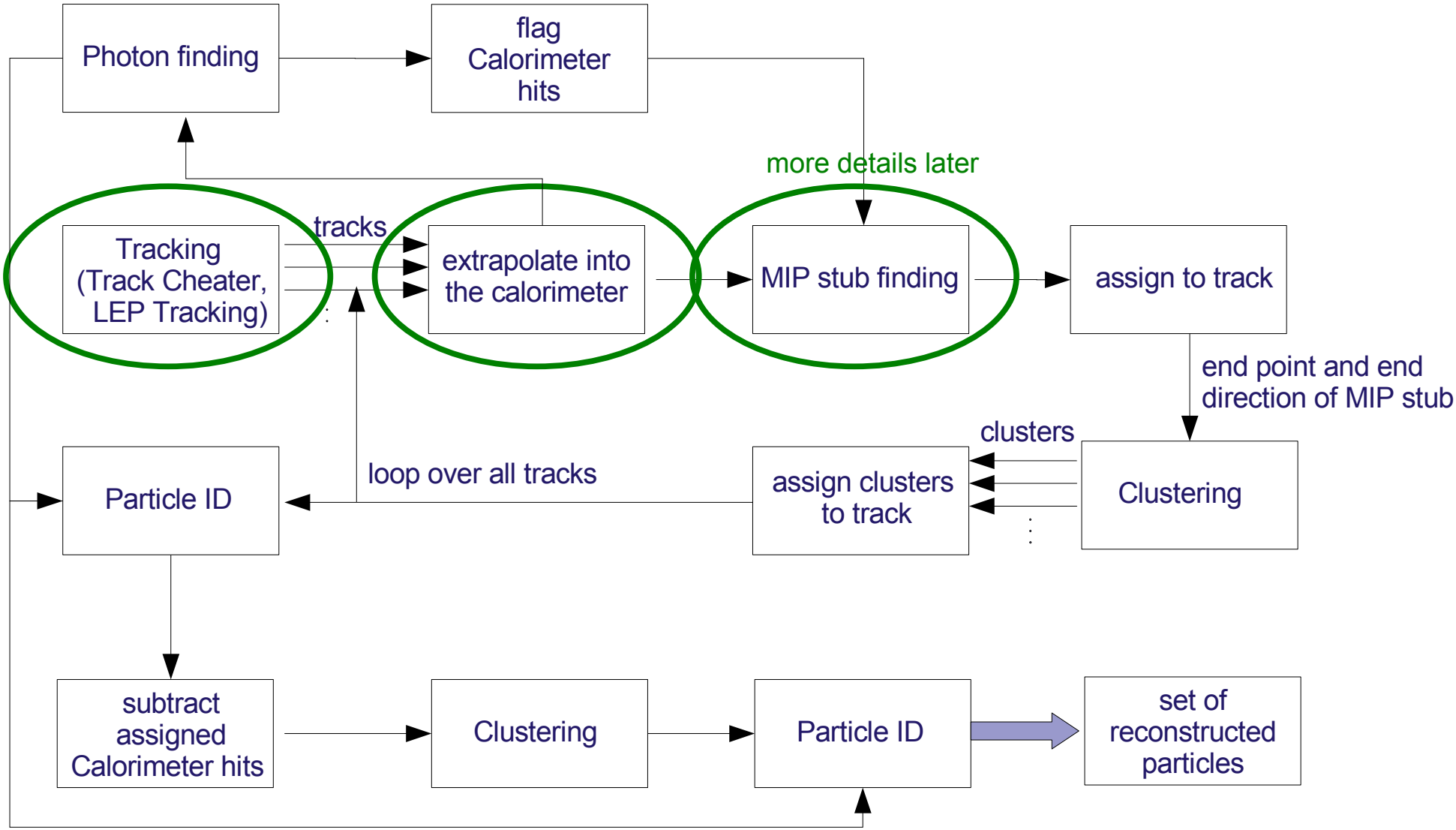
'Track-Based PFlow' in Marlin / Software Chain



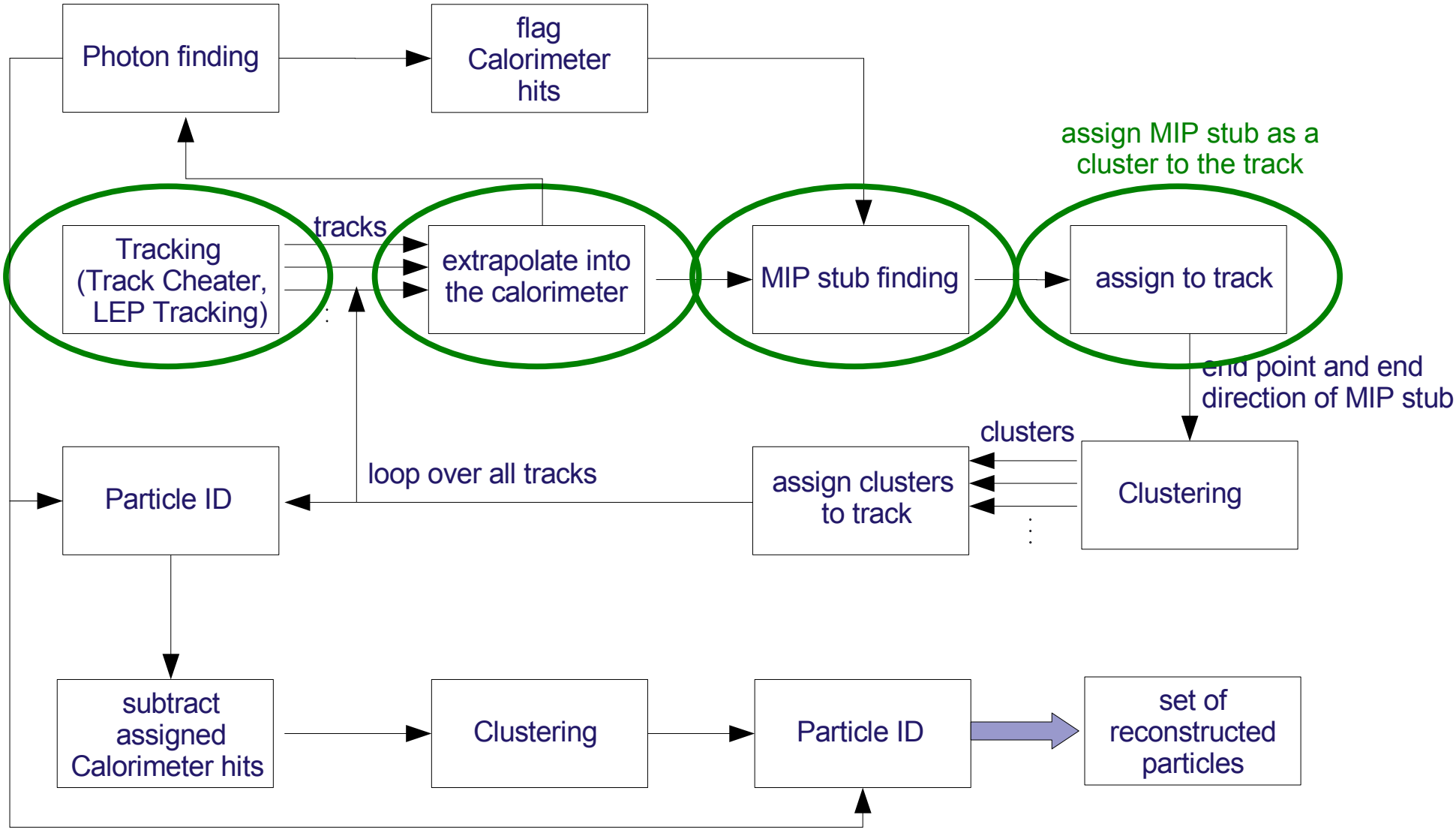
'Track-Based PFlow' in Marlin / Software Chain



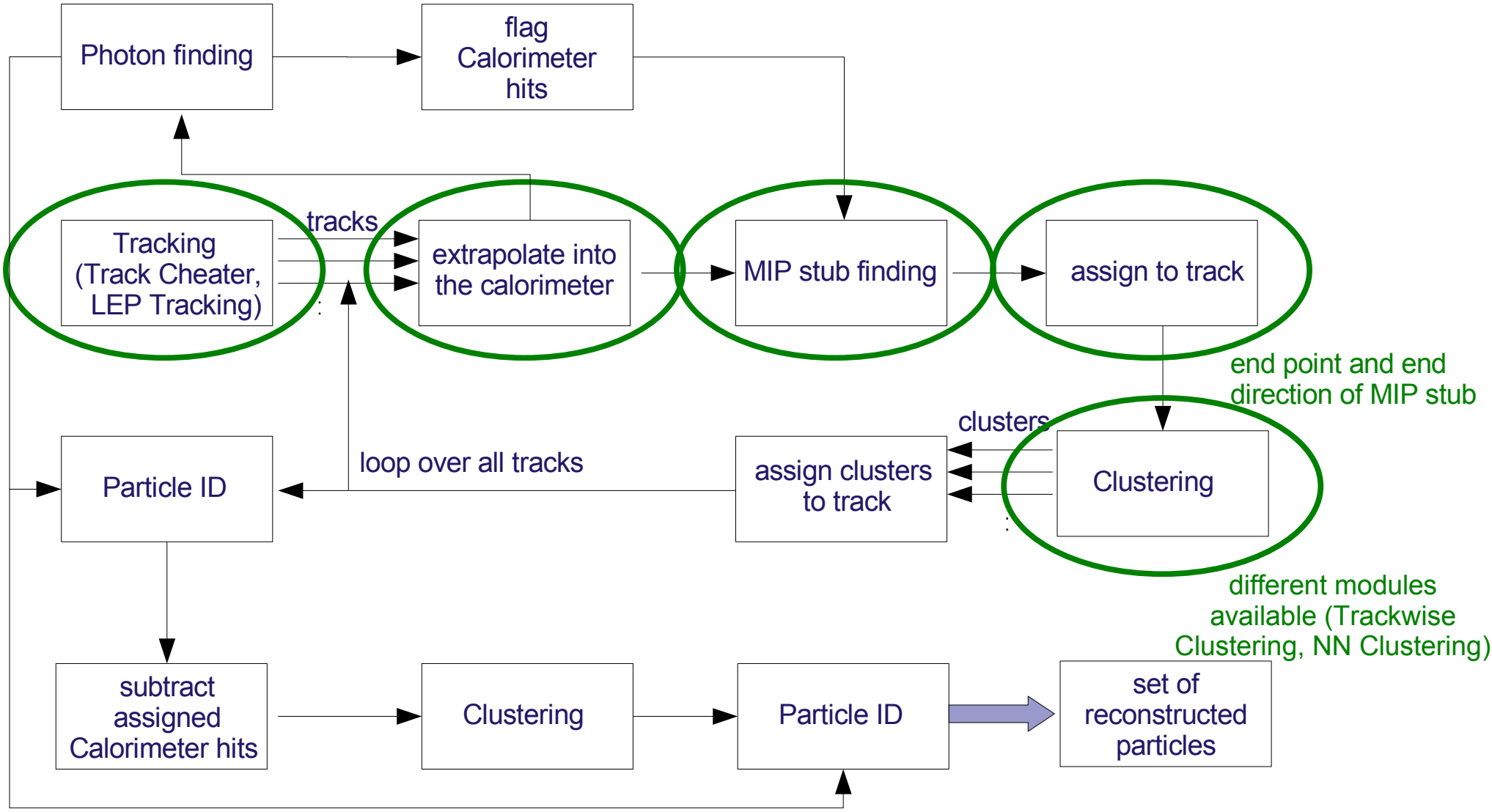
'Track-Based PFlow' in Marlin / Software Chain



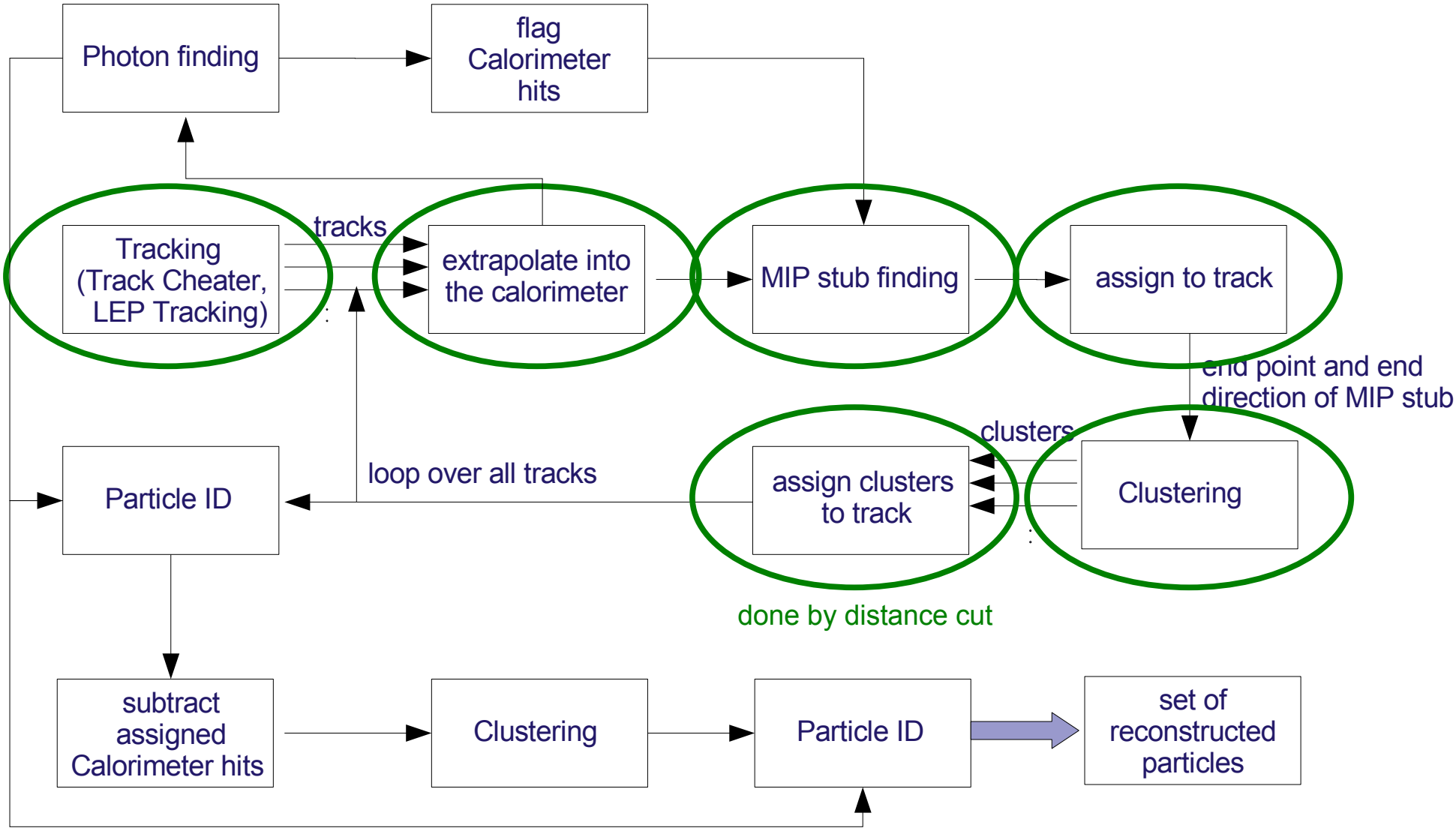
'Track-Based PFlow' in Marlin / Software Chain



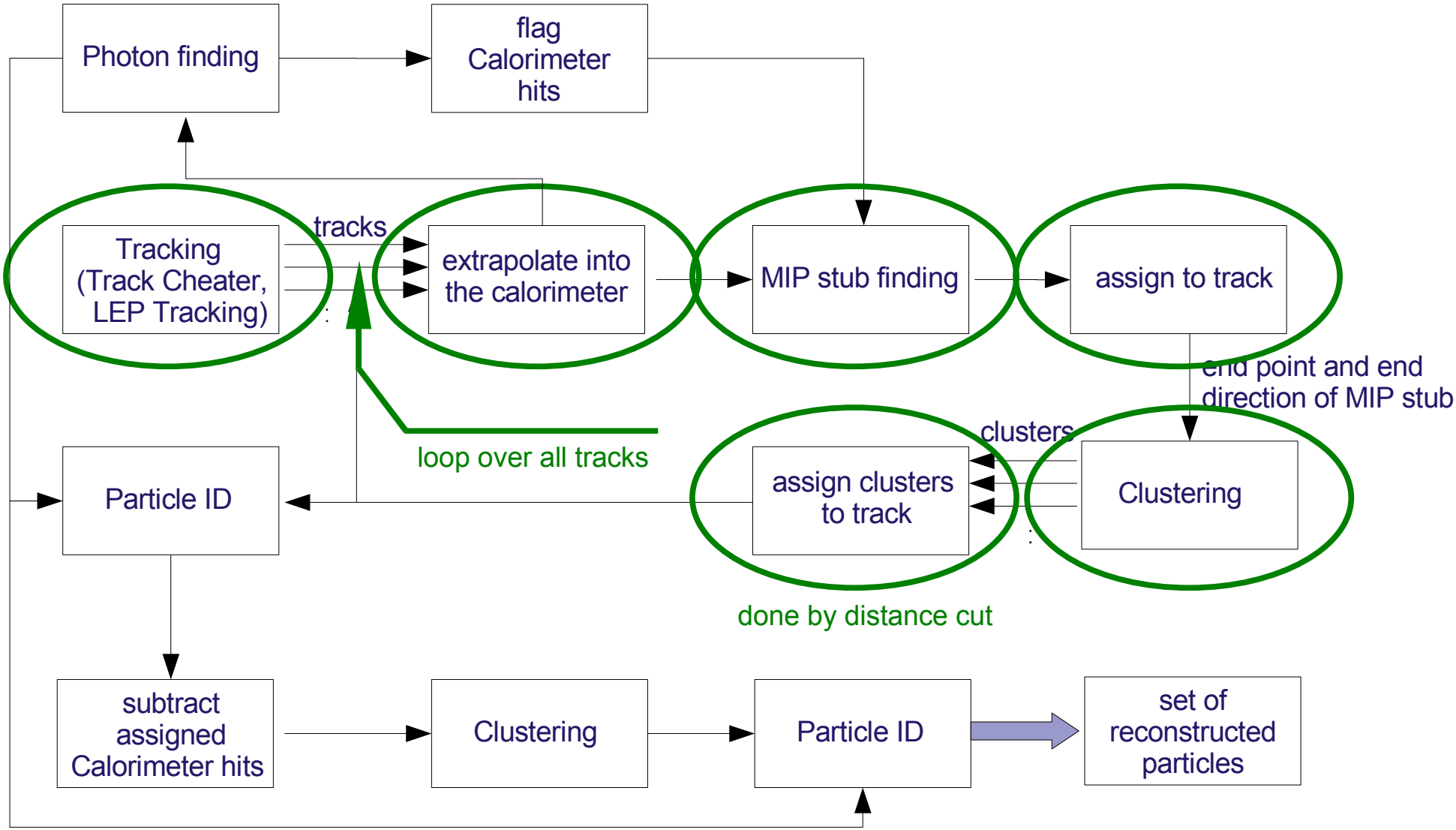
'Track-Based PFlow' in Marlin / Software Chain



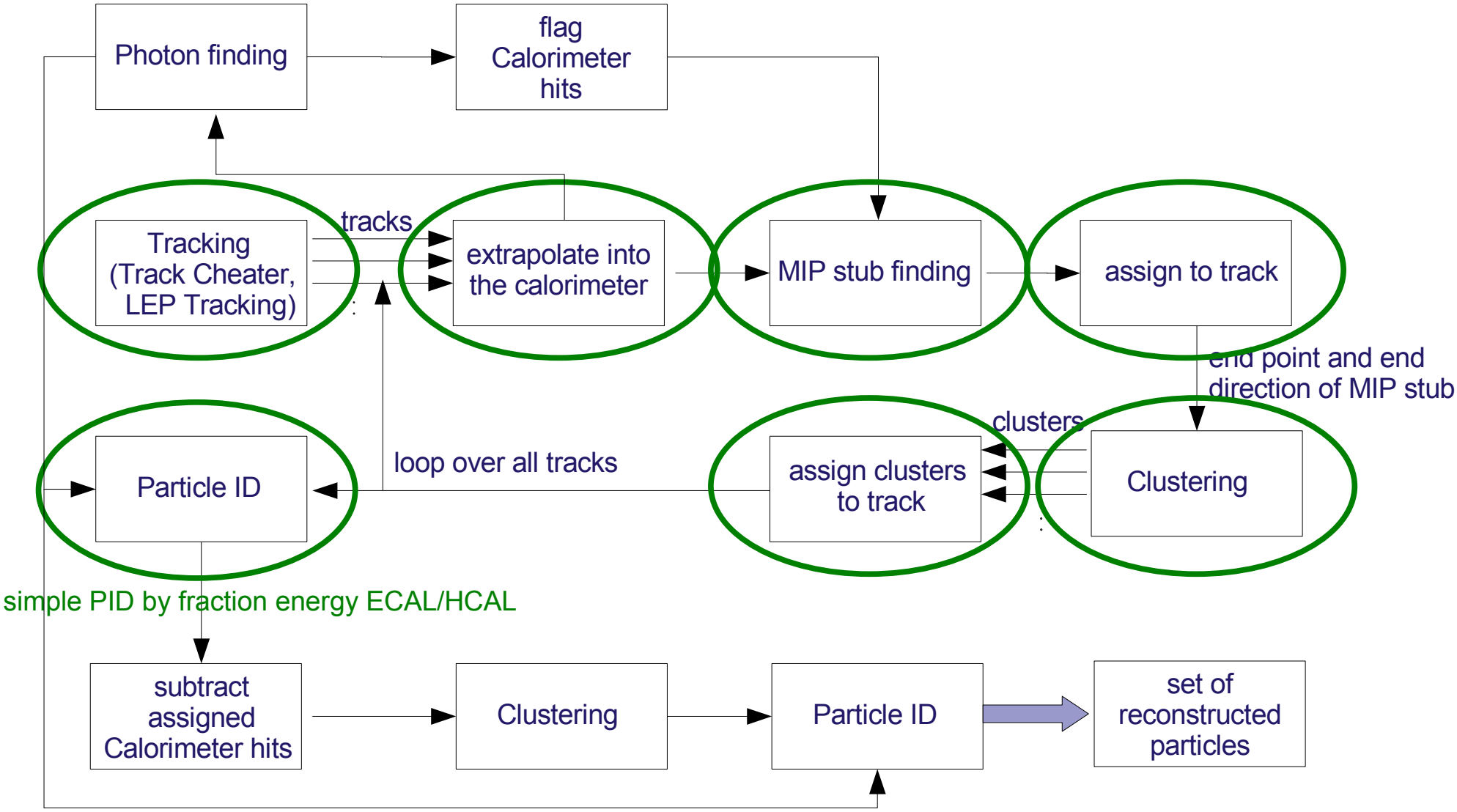
'Track-Based PFlow' in Marlin / Software Chain



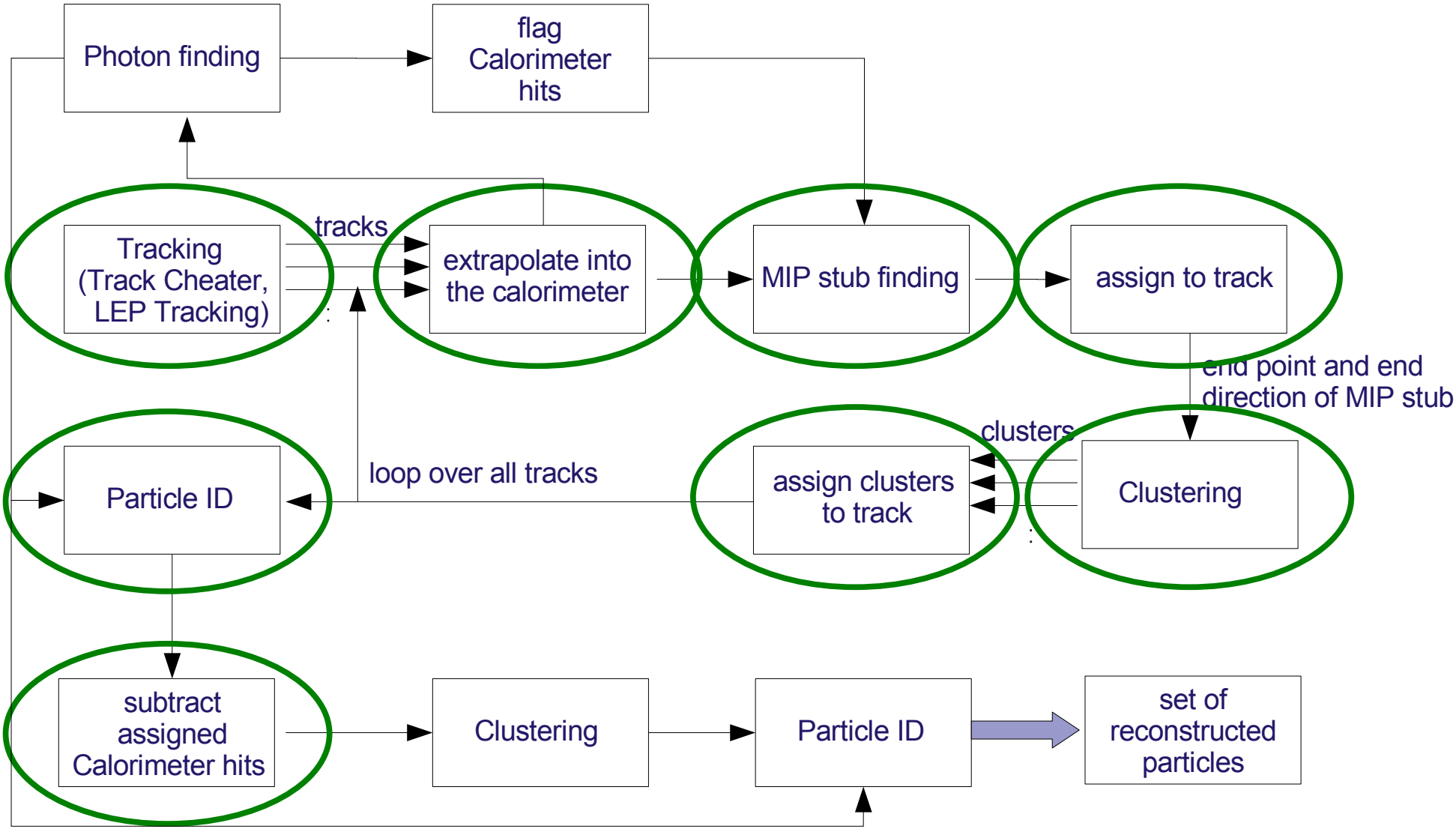
'Track-Based PFlow' in Marlin / Software Chain



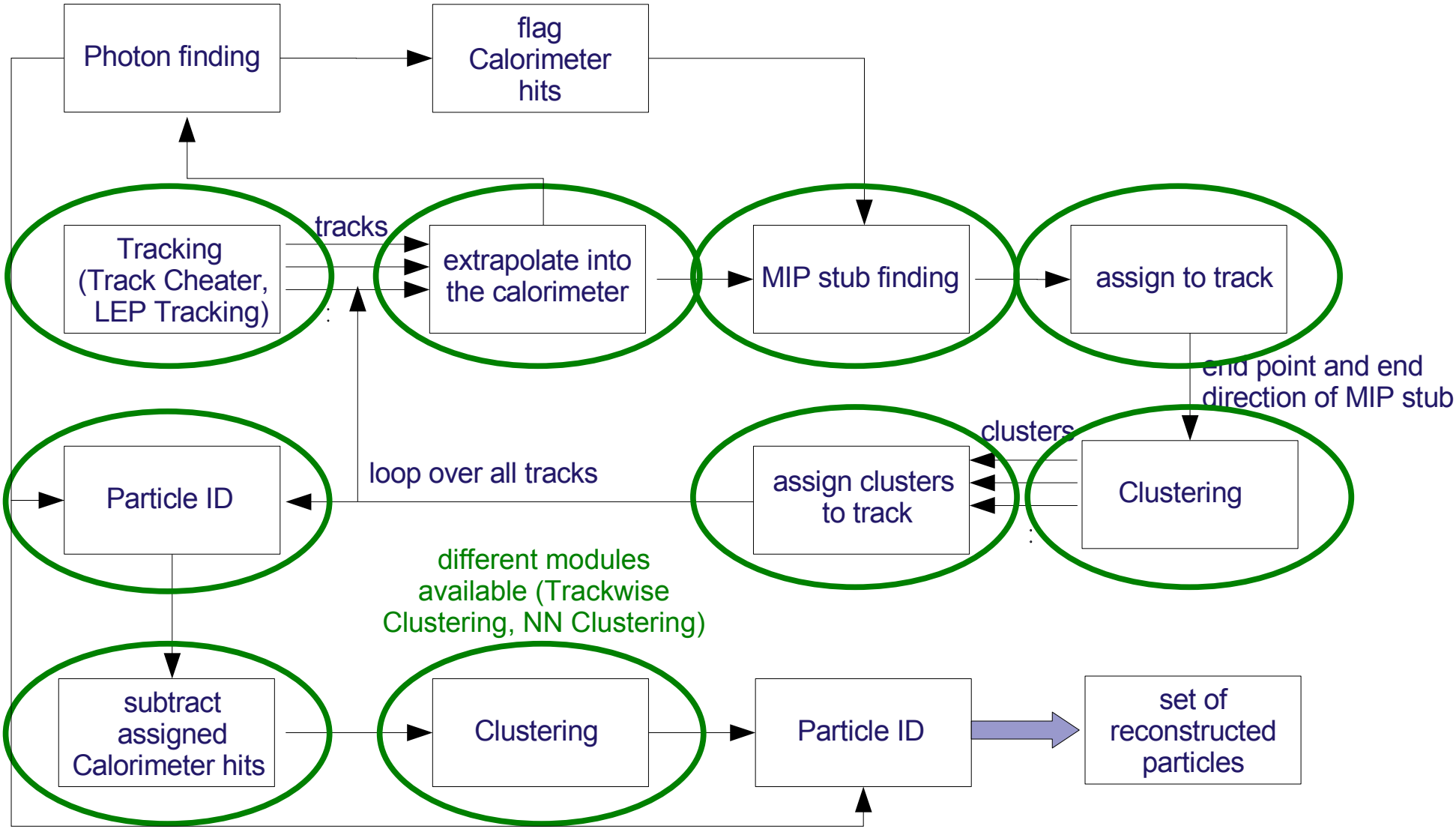
'Track-Based PFlow' in Marlin / Software Chain



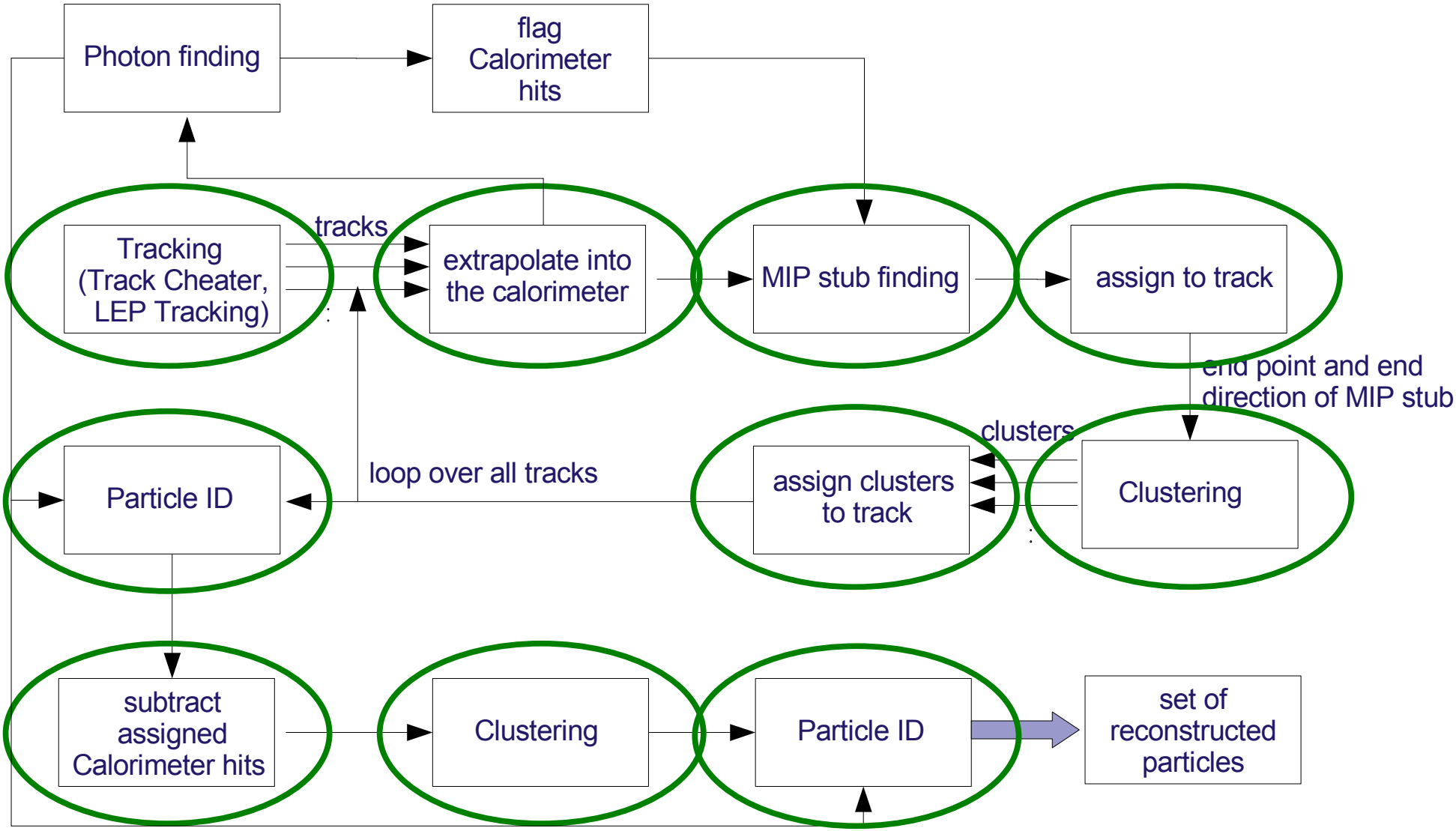
'Track-Based PFlow' in Marlin / Software Chain



'Track-Based PFlow' in Marlin / Software Chain

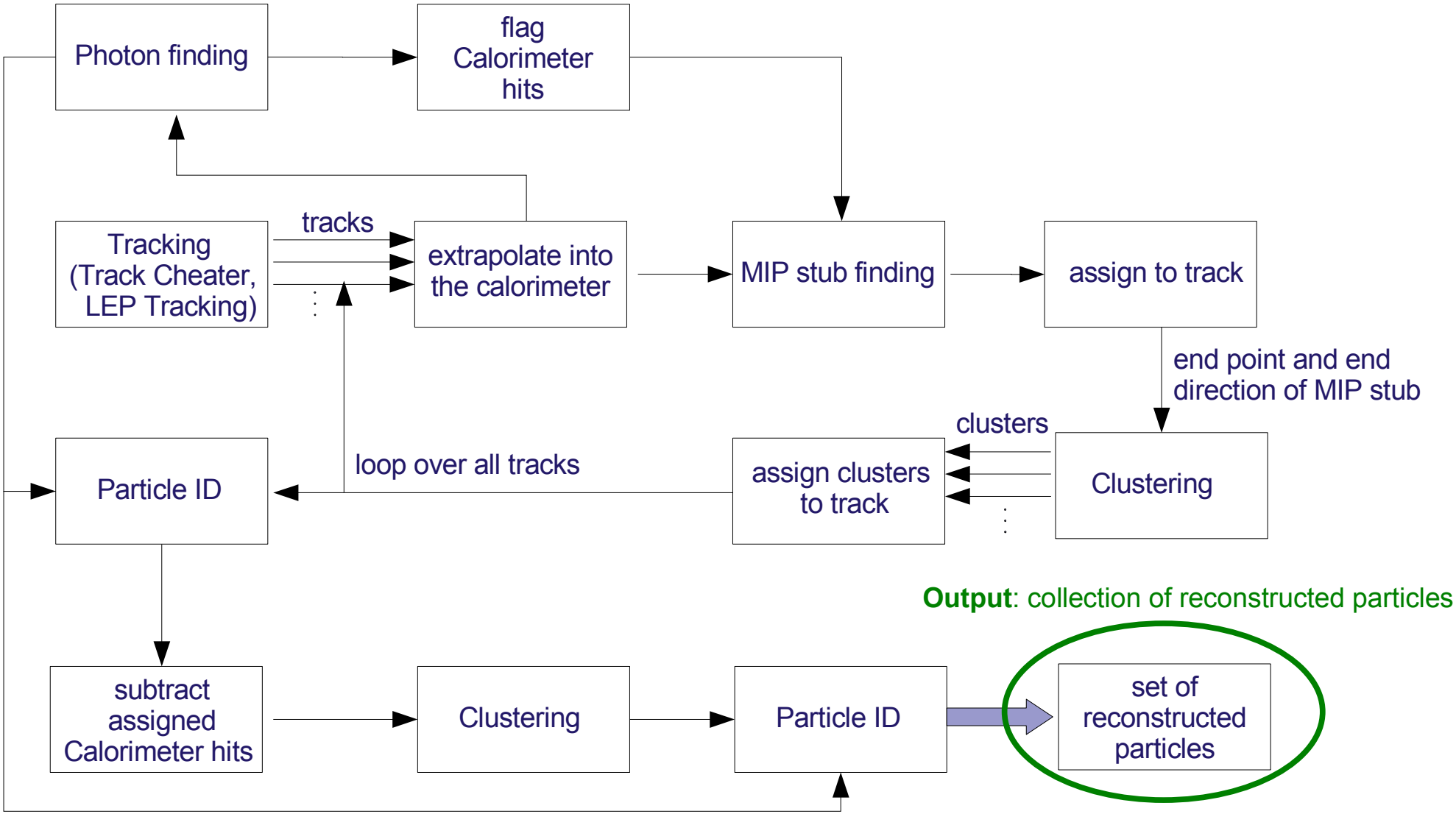


'Track-Based PFlow' in Marlin / Software Chain

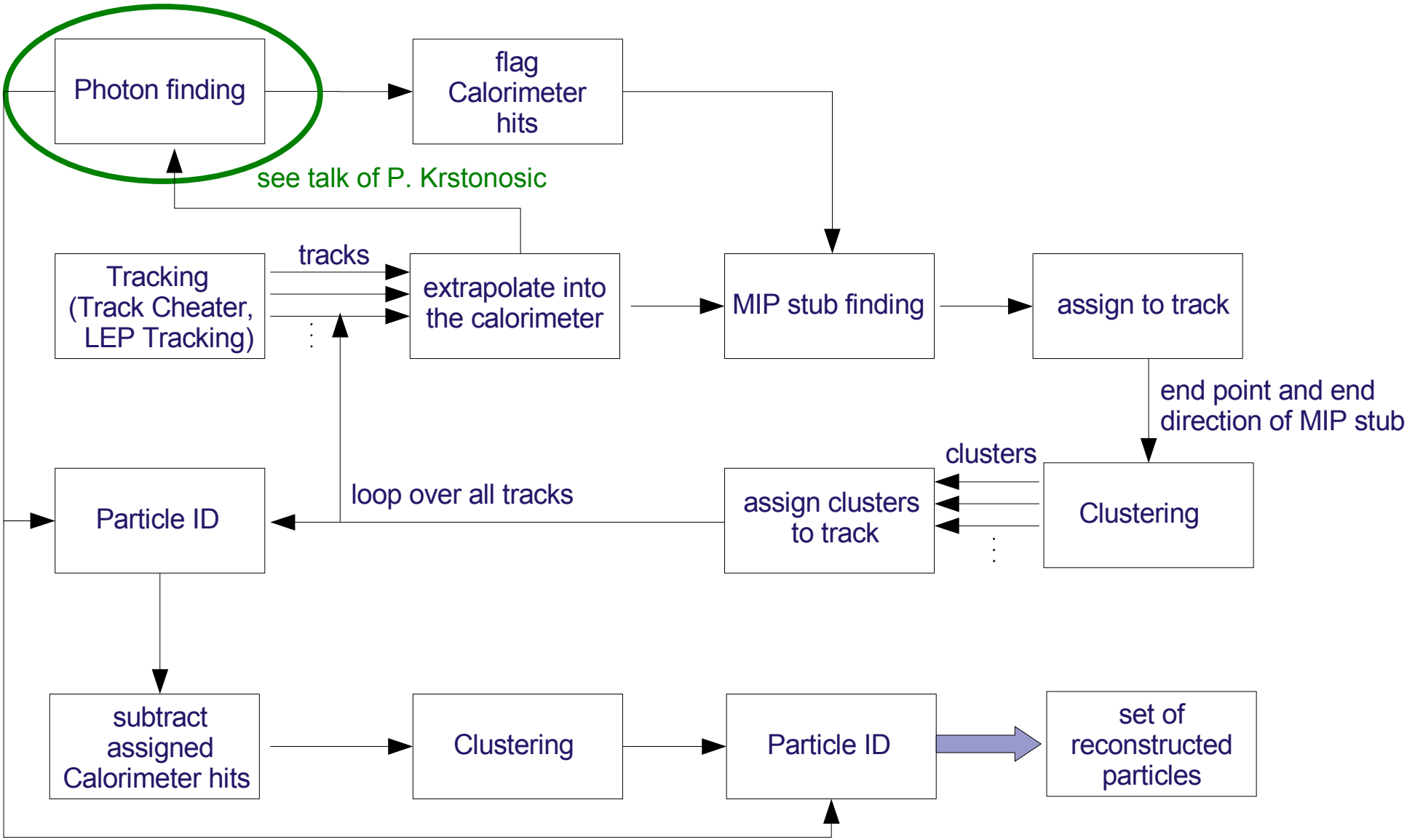


simple PID by fraction energy ECAL/HCAL

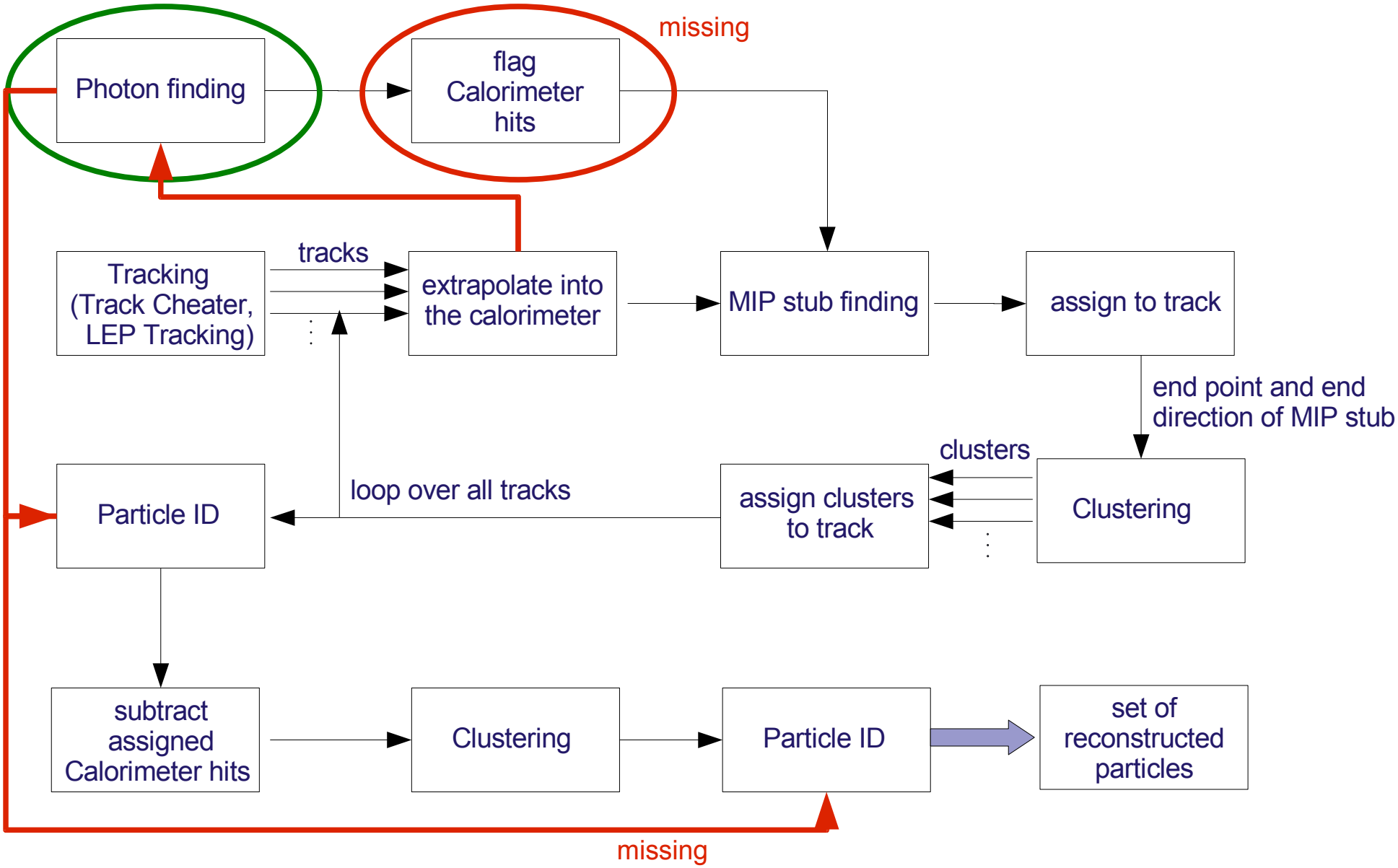
'Track-Based PFlow' in Marlin / Software Chain



'Track-Based PFlow' in Marlin / Software Chain



'Track-Based PFlow' in Marlin / Software Chain



'Track-Based PFlow' in Marlin / Software Chain

- first step towards a **'track-based'** PFlow algorithm in Marlin
  - ➔ full software chain established (output: reconstr. particles)
- ➢ **work in progress** → no results on global variables (ΔE/E) yet
- ➢ code needs cleaning, debugging, documentation and optimisation (release in MarlinReco)
- ➢ enhance modularity, define interfaces
- several modules / interfaces are **missing**:
  - ➔ cluster overlaps (confusion term)
  - ➔ adaptive clustering procedures
  - ➔ refining of reconstruction by iteration of sub-modules (clustering) or the **whole** PFlow chain
  - ➔ ...

your inputs / ideas are welcome

- ➢ some more details about the MIP stub finding:

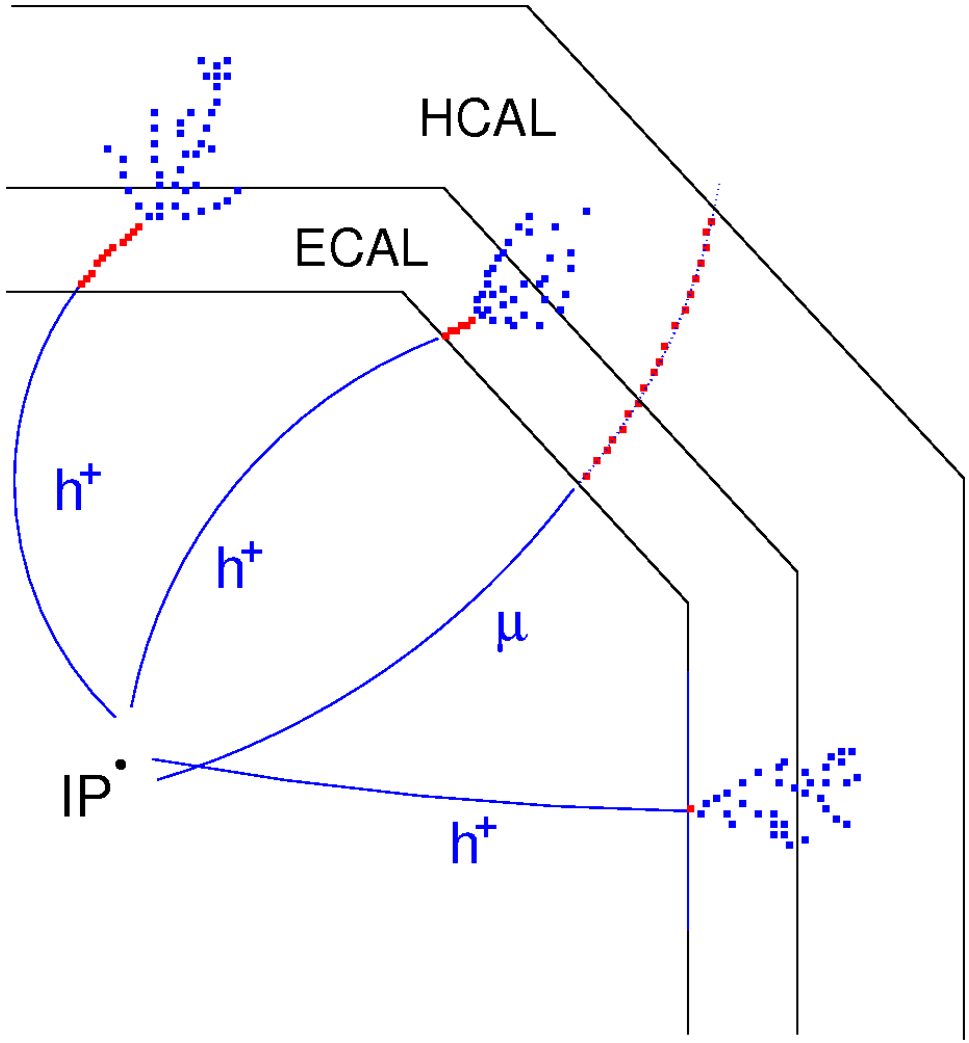
MIP stub finding

characteristics of MIP stubs:

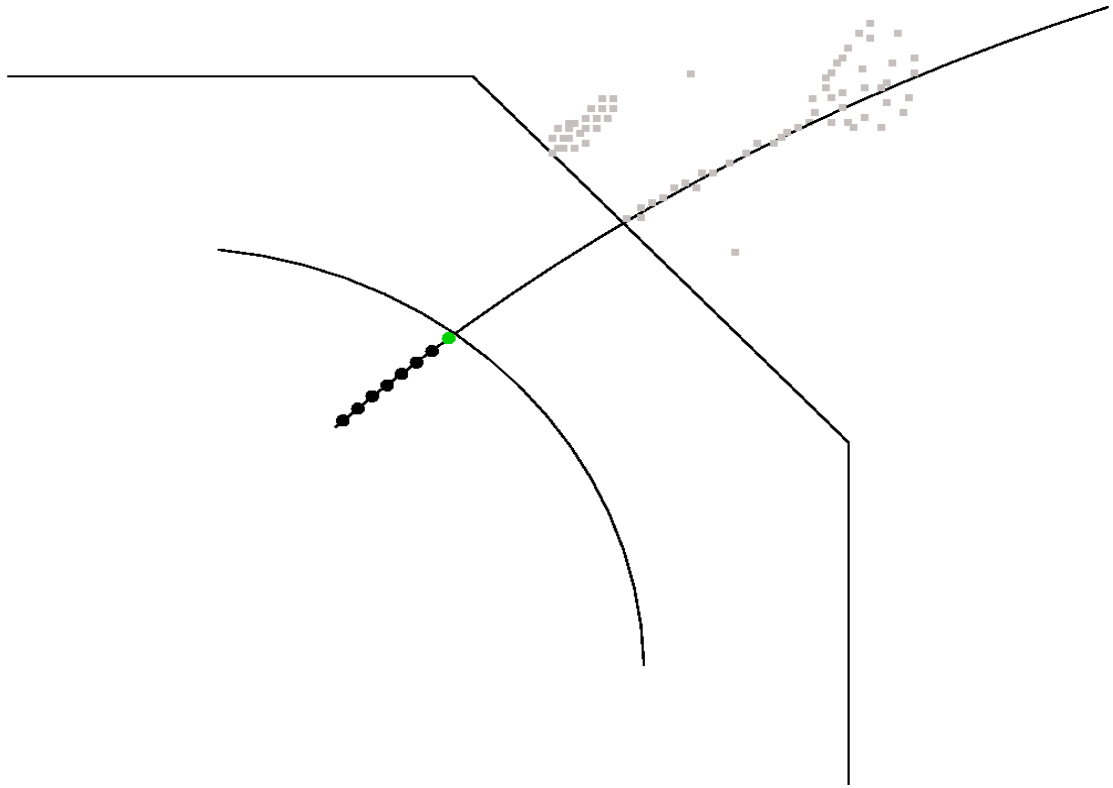
- **energy loss** per path length
- → energy deposition per cell
- → length of track-like segment
  - ➢ find muons as well

algorithm to find MIP stubs:

- **purely topological**: sequence of hits along the extrapolated track
- minimal number of such hits required
- → **extended geometry / material** system needed to take energy deposition into account (GEAR)

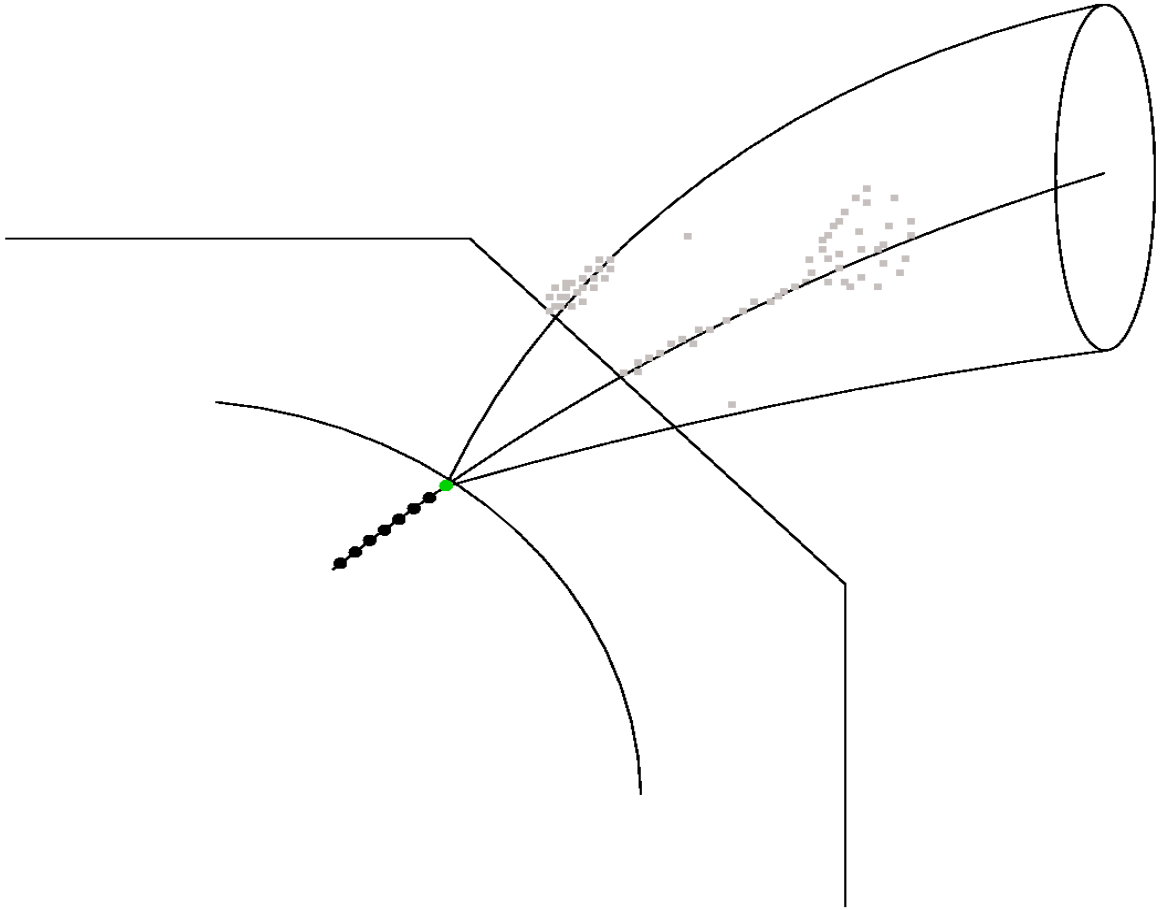


Details of MIP stub finding



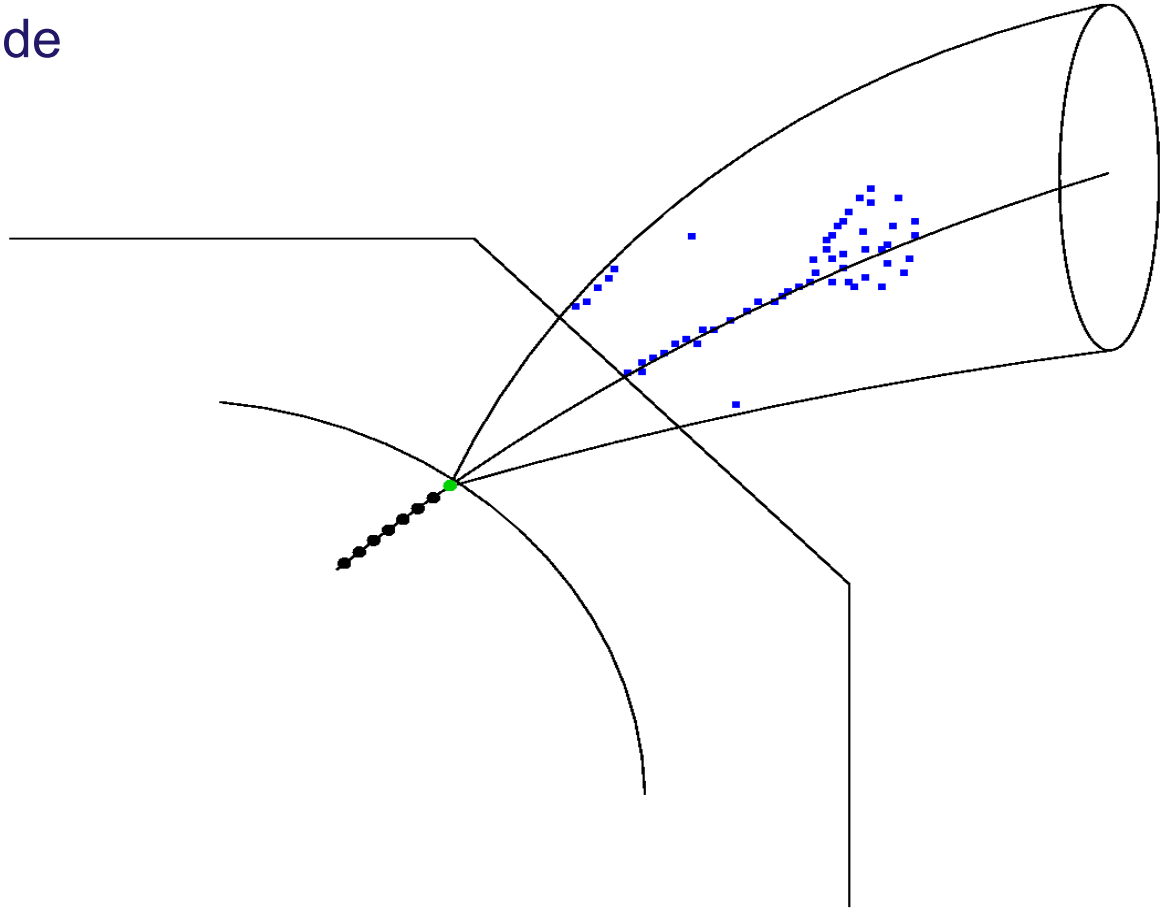
Details of MIP stub finding

- put cone-like tube around extrapolated trajectory



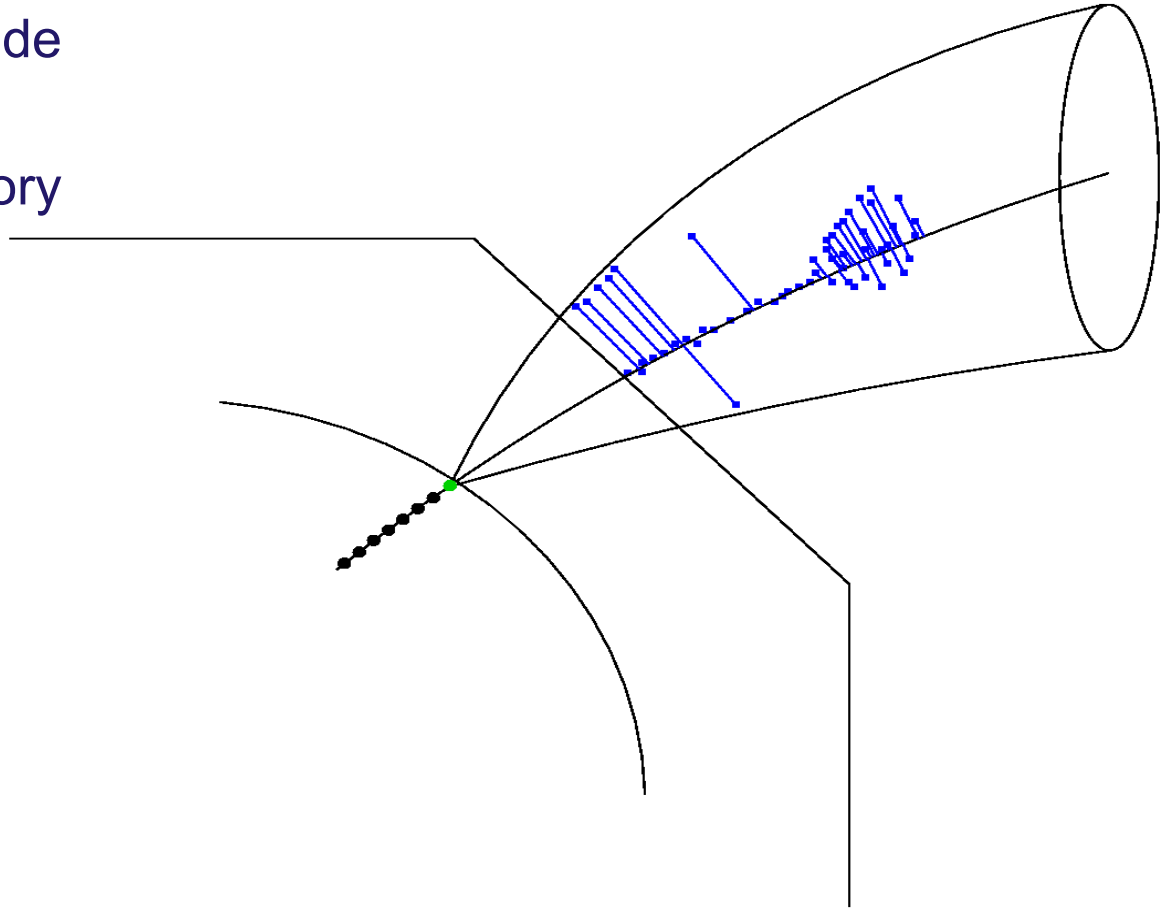
Details of MIP stub finding

- put cone-like tube around extrapolated trajectory
- cut calorimeter hits outside cone-like tube



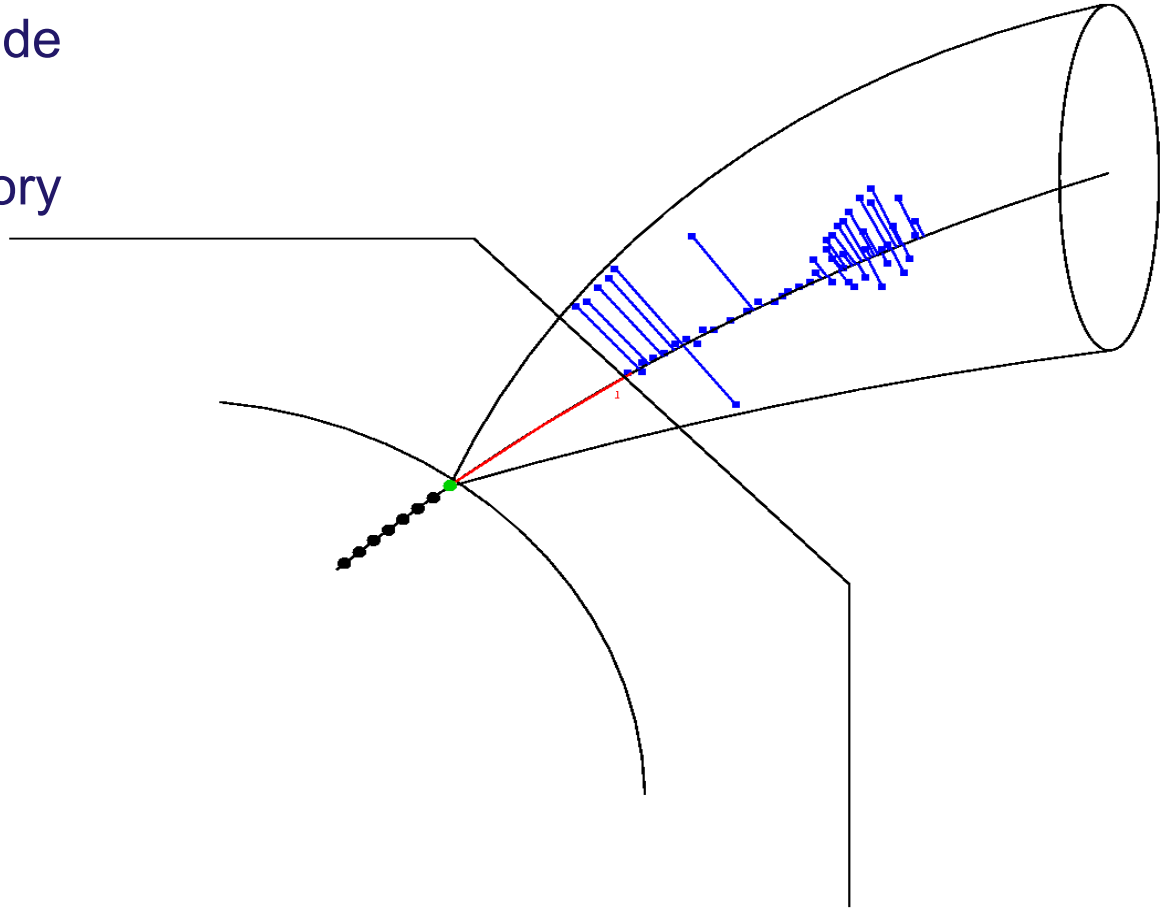
Details of MIP stub finding

- put cone-like tube around extrapolated trajectory
- cut calorimeter hits outside cone-like tube
- project all hits on trajectory



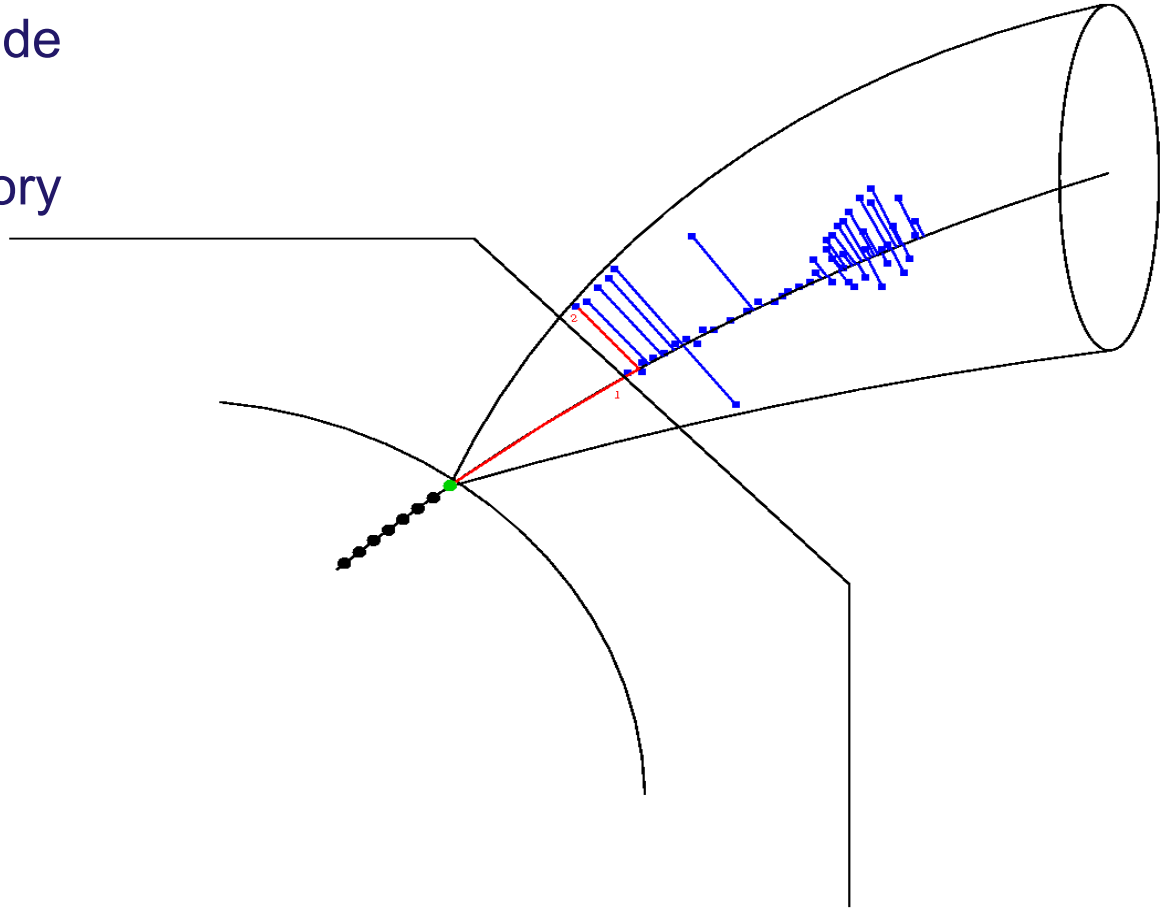
Details of MIP stub finding

- put cone-like tube around extrapolated trajectory
- cut calorimeter hits outside cone-like tube
- project all hits on trajectory
- calculate path length on trajectory for all hits



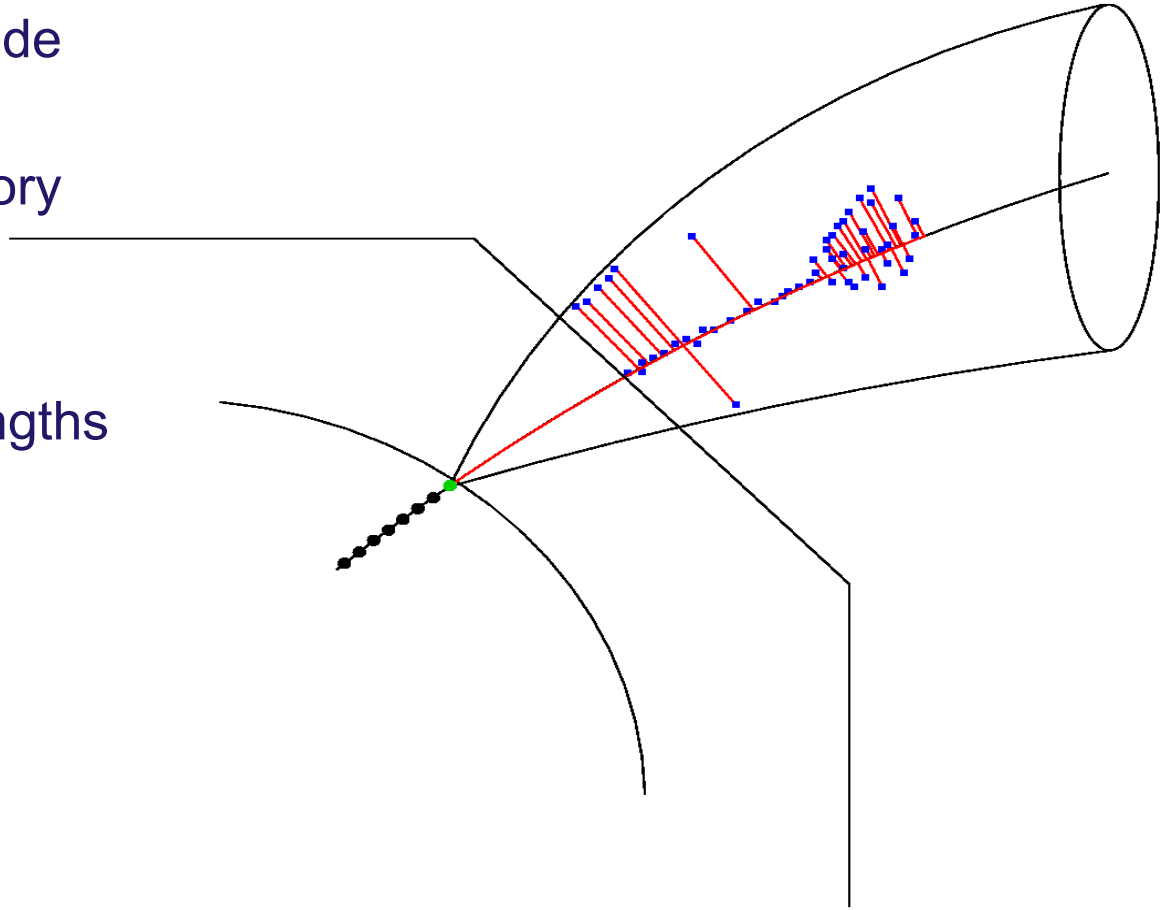
Details of MIP stub finding

- put cone-like tube around extrapolated trajectory
- cut calorimeter hits outside cone-like tube
- project all hits on trajectory
- calculate path length on trajectory for all hits



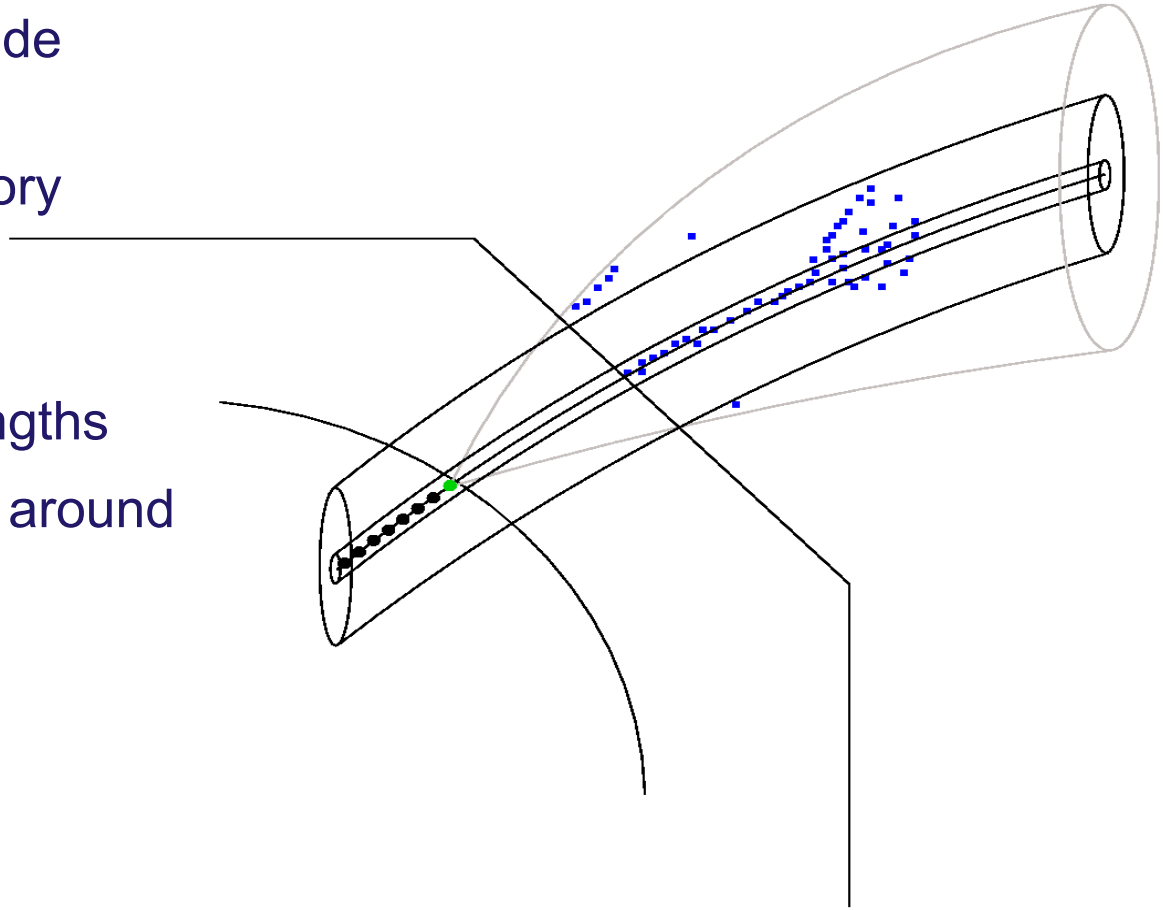
Details of MIP stub finding

- put cone-like tube around extrapolated trajectory
- cut calorimeter hits outside cone-like tube
- project all hits on trajectory
- calculate path length on trajectory for all hits
- sort hits by their path lengths



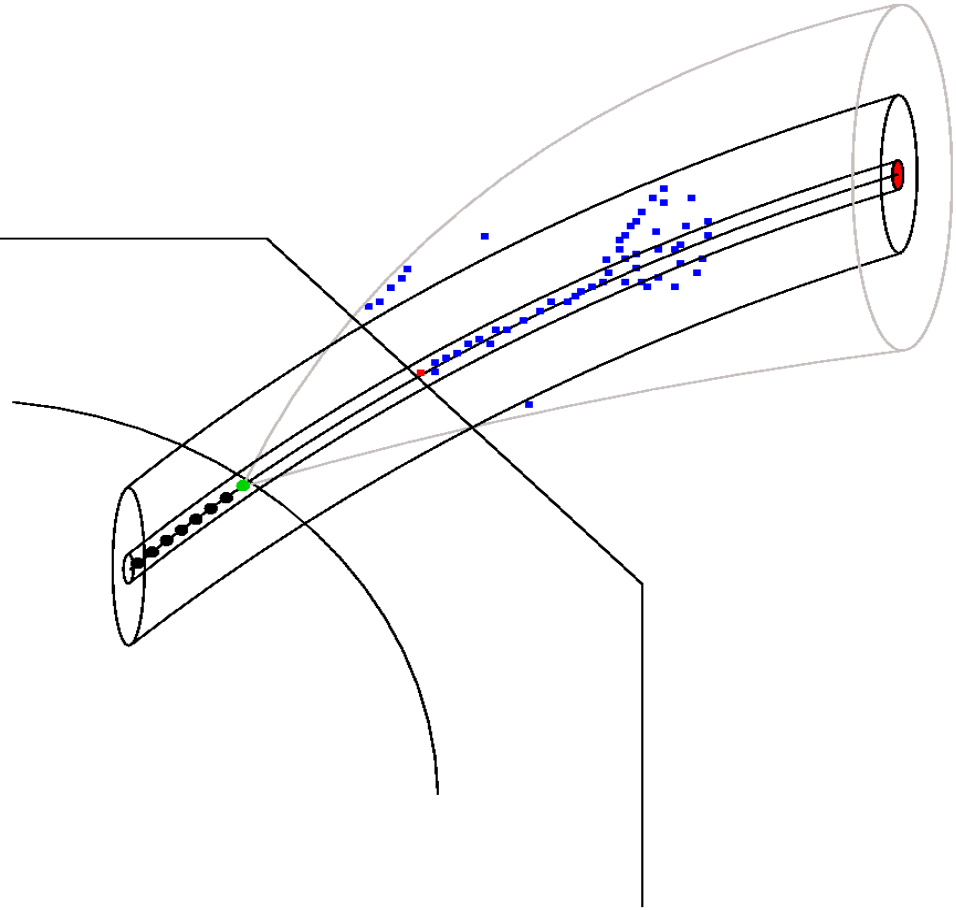
Details of MIP stub finding

- put cone-like tube around extrapolated trajectory
- cut calorimeter hits outside cone-like tube
- project all hits on trajectory
- calculate path length on trajectory for all hits
- sort hits by their path lengths
- put two cylindrical tubes around extrapolated trajectory



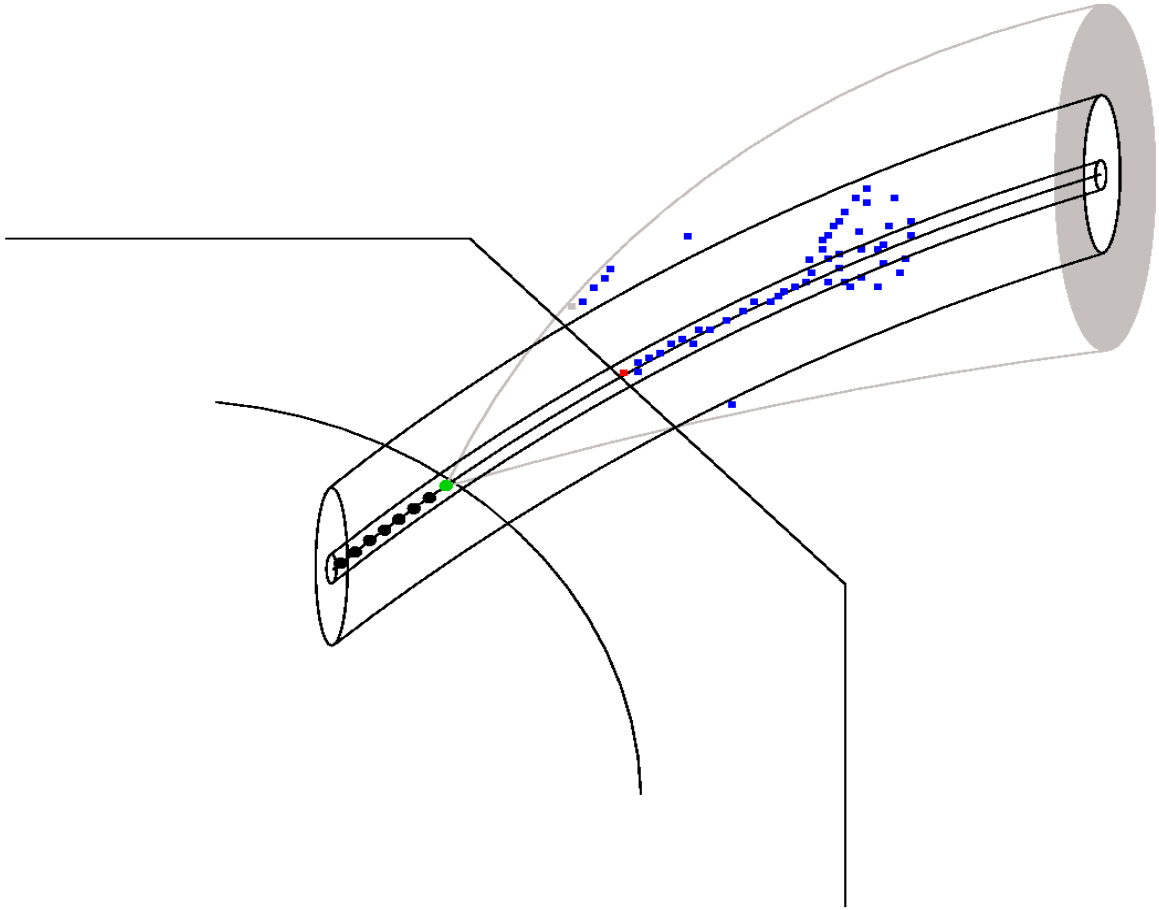
Details of MIP stub finding

- put cone-like tube around extrapolated trajectory
- cut calorimeter hits outside cone-like tube
- project all hits on trajectory
- calculate path length on trajectory for all hits
- sort hits by their path lengths
- put two cylindrical tubes around extrapolated trajectory
- take first hit according to its path length and add it to the MIP stub if it is located inside the inner cylindrical tube



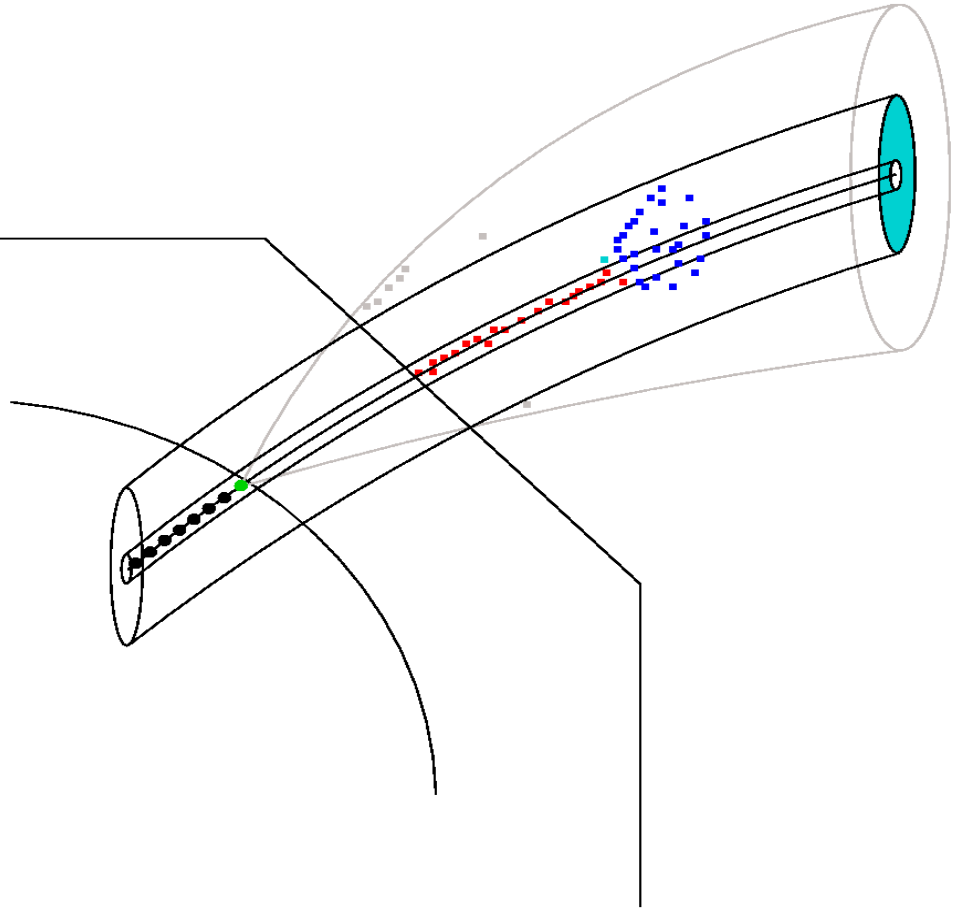
Details of MIP stub finding

- take the next hit and discard it if it is located outside the outer tube



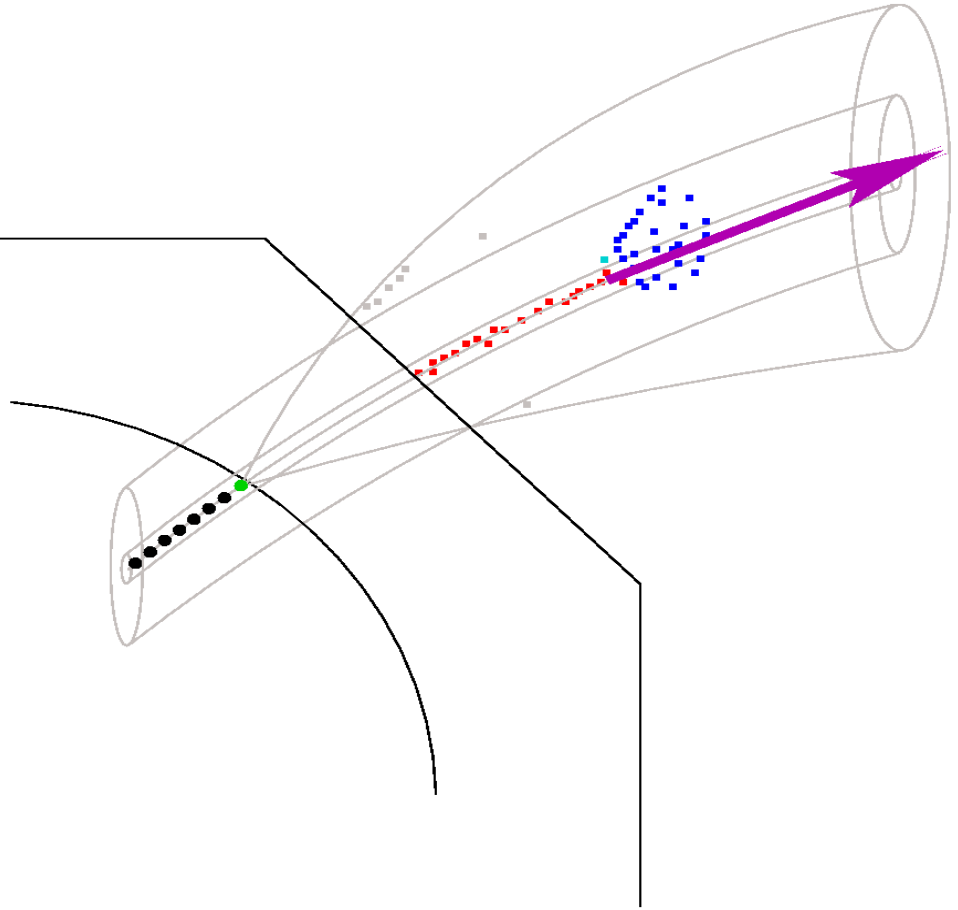
Details of MIP stub finding

- take the next hit and discard it if it is located outside the outer tube
- repeat this procedure for all hits until a hit outside the inner and inside the outer cylinder tube is found ('veto-cylinder')



Details of MIP stub finding

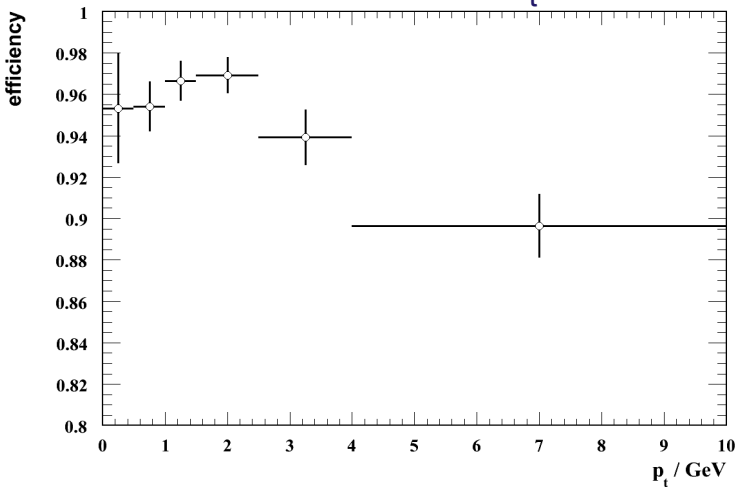
- take the next hit and discard it if it is located outside the outer tube
- repeat this procedure for all hits until a hit outside the inner and inside the outer cylinder tube is found ('veto-cylinder')
- stop the MIP stub finding
- take the projection of the last hit collected for the MIP stub as a start point for clustering
- take the direction (tangent) of this point as a start direction for clustering

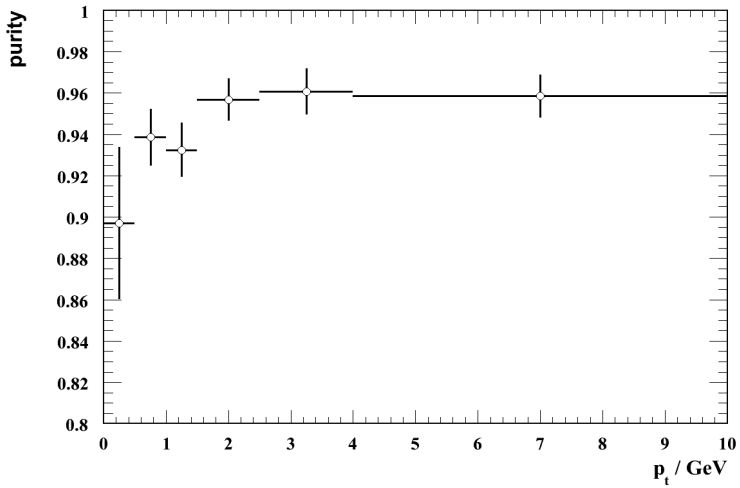


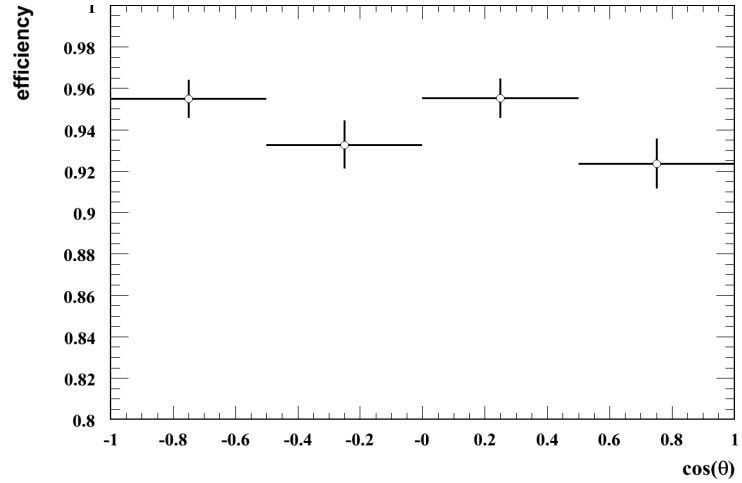
Details of MIP stub finding

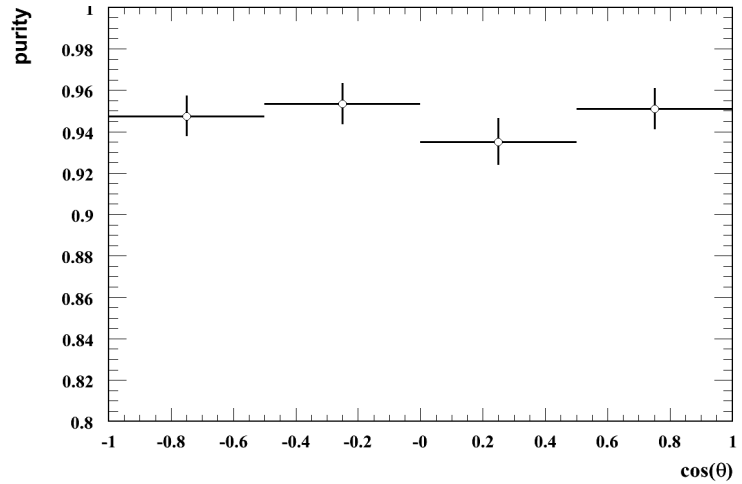
first results for $Z^0 \rightarrow$ uds @ 91.2 GeV, LDC00Sc R(1690mm) L(2730mm)

- overall efficiency >90%, overall purity >90%
- efficiency and purity vs. p_t and cos(θ):









Summary and Outlook

- ✔ first step towards a **'track-based'** PFlow algorithm in Marlin
- ✔ first studies on modules have been performed
- ✗ studies on **global performance** (ΔE/E) missing
- a lot more work needs to be done
  - → modules / interfaces / structure
  - → clean up code
  - → release in MarlinReco

your inputs / ideas are welcome