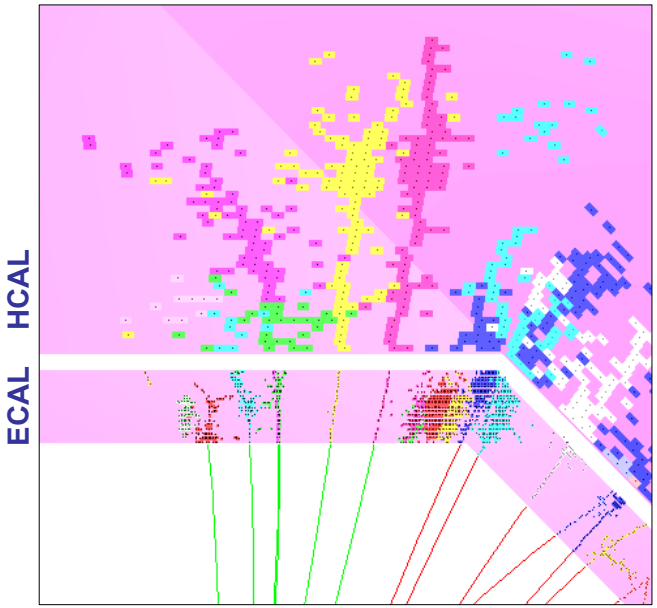
PandoraPFA and LArSoft

Mark Thomson, Andy Blake and John Marshall

What is Pandora?

★ **PandoraPFA is a lightweight framework for pattern recognition + a toolkit of pattern recognition algorithms**

- **Developed for high granularity at the ILC** (M. Thomson)
- **Re-designed as a flexible framework** (J.Marshall, M.Thomson)
- **Now used for calorimeter reconstruction in almost all ILC physics studies**



★ **LC calorimetry:**

- **Large number of hits**
- **Tracks and showers in a dense medium**

Looks familiar ?

Why use it?

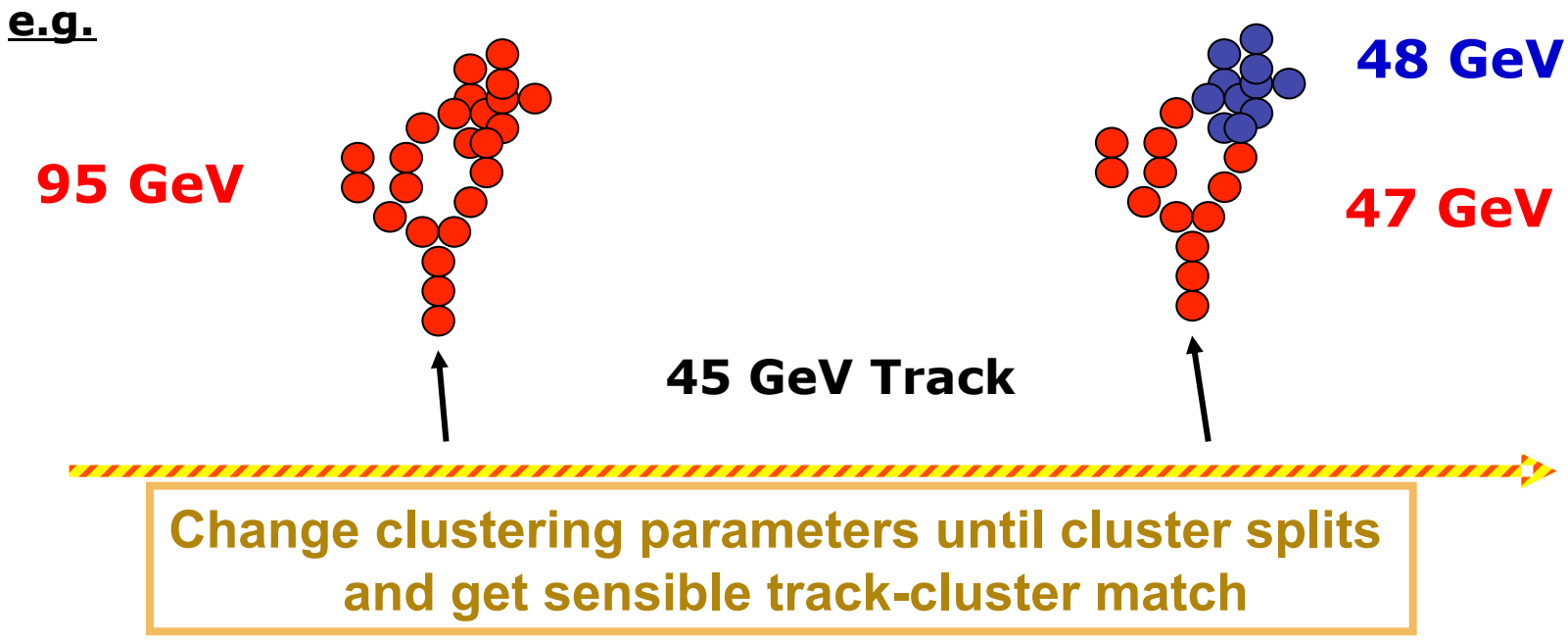
- **PandoraPFA is designed as a framework for pattern recognition**
  - It is not a generic framework such as Gaudi, LArSoft, …
    - Limited detector geometry
    - Limited persistency model

- **Why not do everything in LArSoft ?**
  - Perfectly possible
  - But for complex pattern recognition problems likely to be easier in
  - Pandora
    - CPU/memory optimised data management
    - Powerful "Archetype" algorithms where Pandora does the "heavy lifting"

e.g. reclustering

★ PandoraPFA easily handles, deeply-nested, highly-iterative operations, e.g. reclustering

★ Collider example: If track and cluster energy inconsistent : RECLUSTER

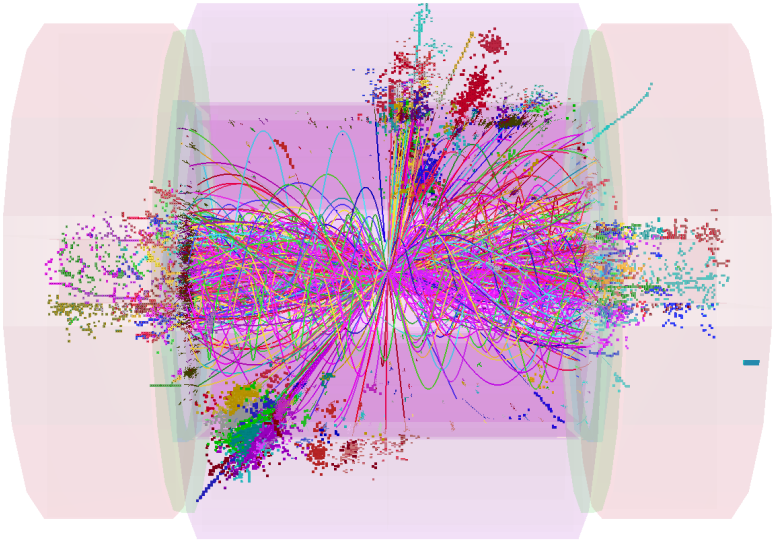


★ Hits can exist (temporarily in multiple clusters) before a decision is made…

Software engineering

- **Pandora Software**
  - Originally written in "physicist C++" =
  - Then thrown away….

- **PandoraNew Software**
  - 6 – 12 months of careful design
  - Robust, fast, optimised container choices, etc.
  - Survived all that has been thrown at it



e.g. e^+e^- physics at 3 TeV CLIC

So want is Pandora?

★ **PandoraPFA is a lightweight framework for pattern recognition + a toolkit of pattern recognition algorithms**

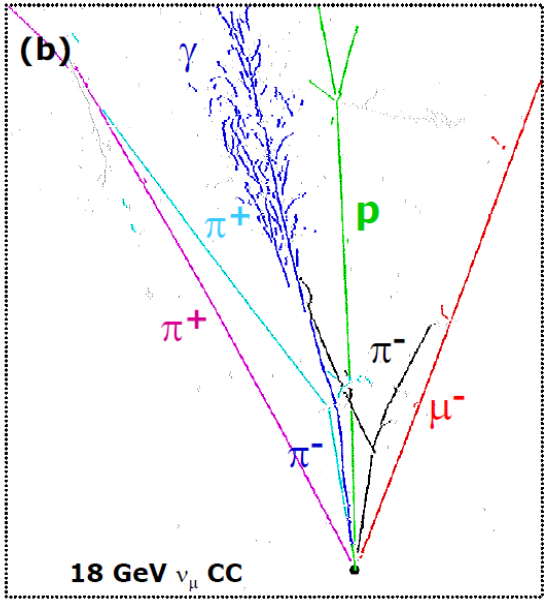
- Runs alongside any reconstruction framework
- Feed native objects (e.g. LArSoft hits) into Pandora
- Once inside Pandora - stored as self-describing Pandora hits
- Within reco framework, write interface to Pandora

★ **First LAr neutrino application developed for LBNE**

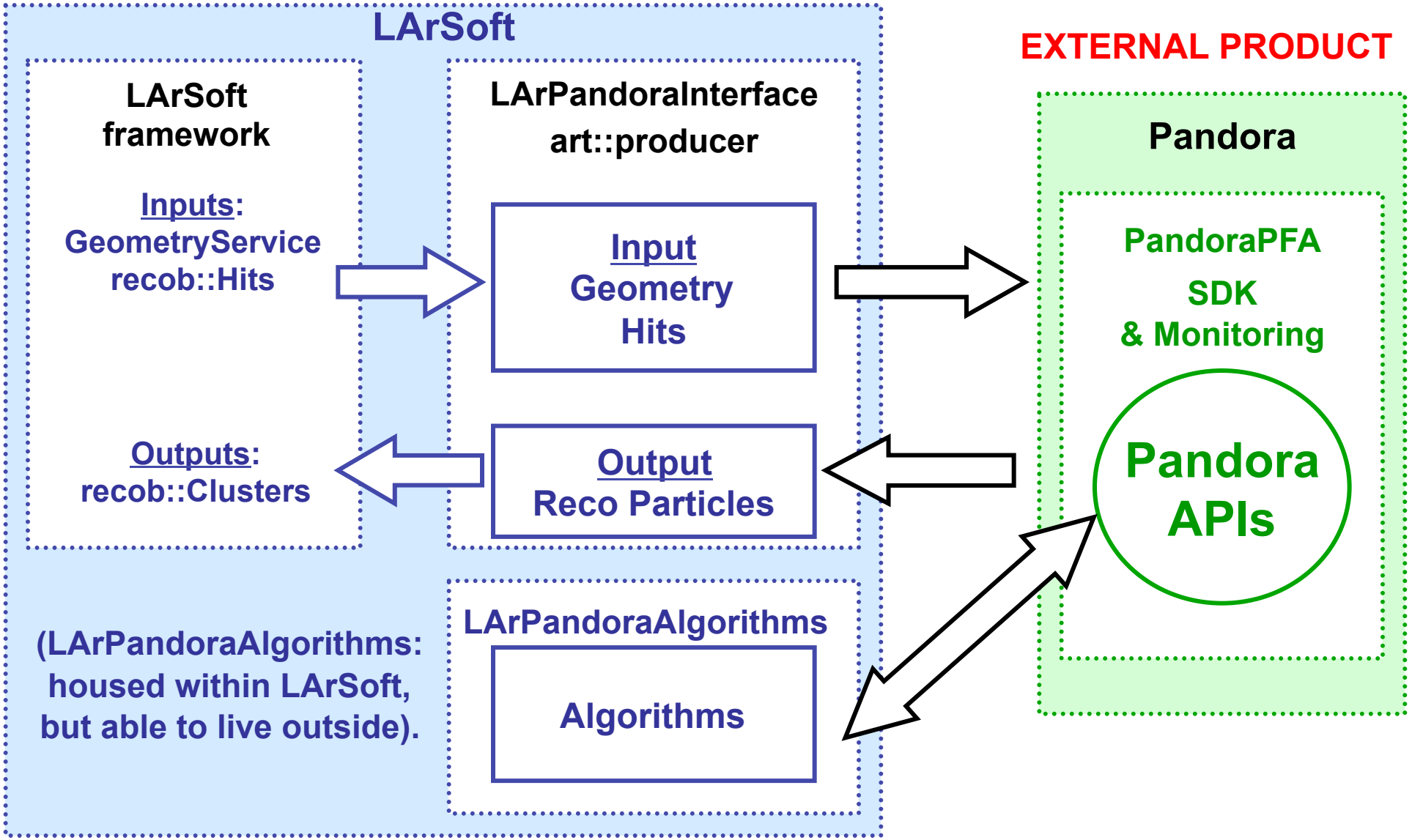
- Actually MicroBooNE simulation with LBNE beam spectrum
- Initial results promising

Focus on LArSoft producer

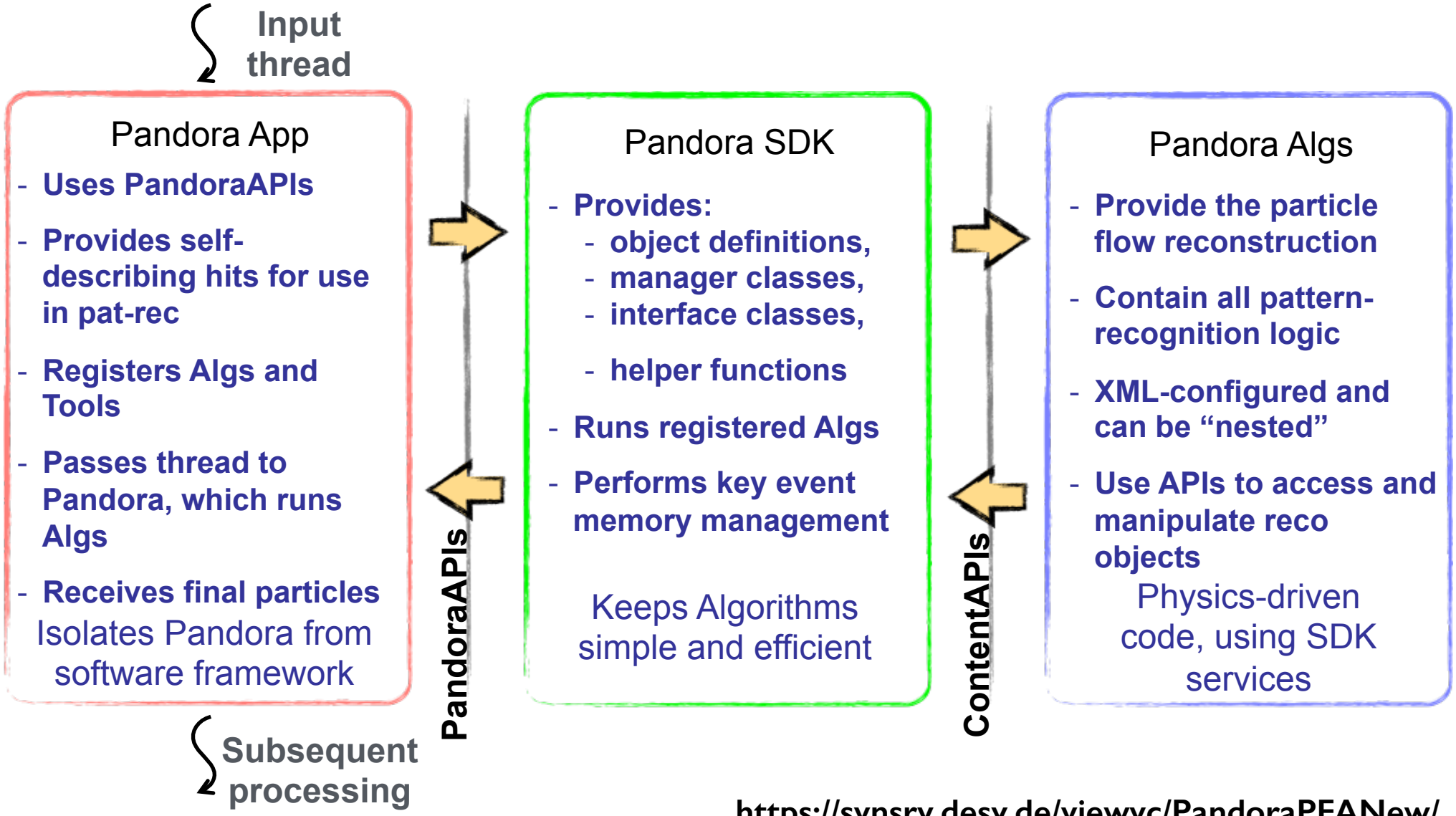
A. Blake, J. Marshall, M. Thomson



Relation to LArSoft

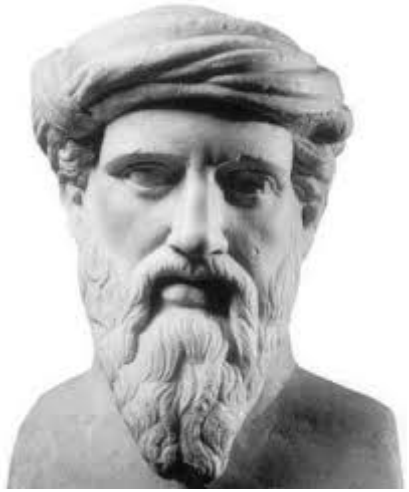


Another way of looking at this



https://svnsrv.desy.de/viewvc/PandoraPFANew/

Reco. Philosophy



- ★ In principle, PandoraPFA could be used to implement any clustering algorithm
- ★ This is not what is done…
- ★ Try to follow a particular philosophy
- ★ This was a big part in the success for ILC:
  - First full demonstration of Pflow at ILC

DOs and and DON'Ts

- ★ NO to: single monolithic algorithms
  - e.g. unlikely single clustering approach covers all topologies
- ★ YES to: many smaller algorithms
  - designed to address specific topologies
  - designed not to make mistakes, undoing mistakes can be hard
- ★ YES to: design before coding
  - think about speed in early on in design and in implementation

LAr: To 3D or not to 3D

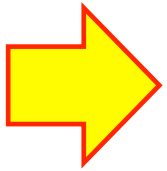
- ★ **Major question: when to go 3D ?**
- ★ **Possibilities**
  - **As early as possible**
    - **PRO: this is what we would like to do**
    - **CON: there will be ambiguities, will make mistakes**
  - **As late as possible**
    - **PRO: we are dealing with inherently 2D readout. x vs time**
    - **CON: in some views cannot resolve particles, will make mistakes**

To 3D or not to 3D

- ★ **Major question: when to go 3D ?**
- ★ **Possibilities**
  - **As early as possible**
    - **PRO: this is what we would like to do**
    - **CON: there will be ambiguities, will make mistakes**
  - **As late as possible**
    - **PRO: we are dealing with inherently 2D readout. x vs time**
    - **CON: in some views cannot resolve particles, will make mistakes**

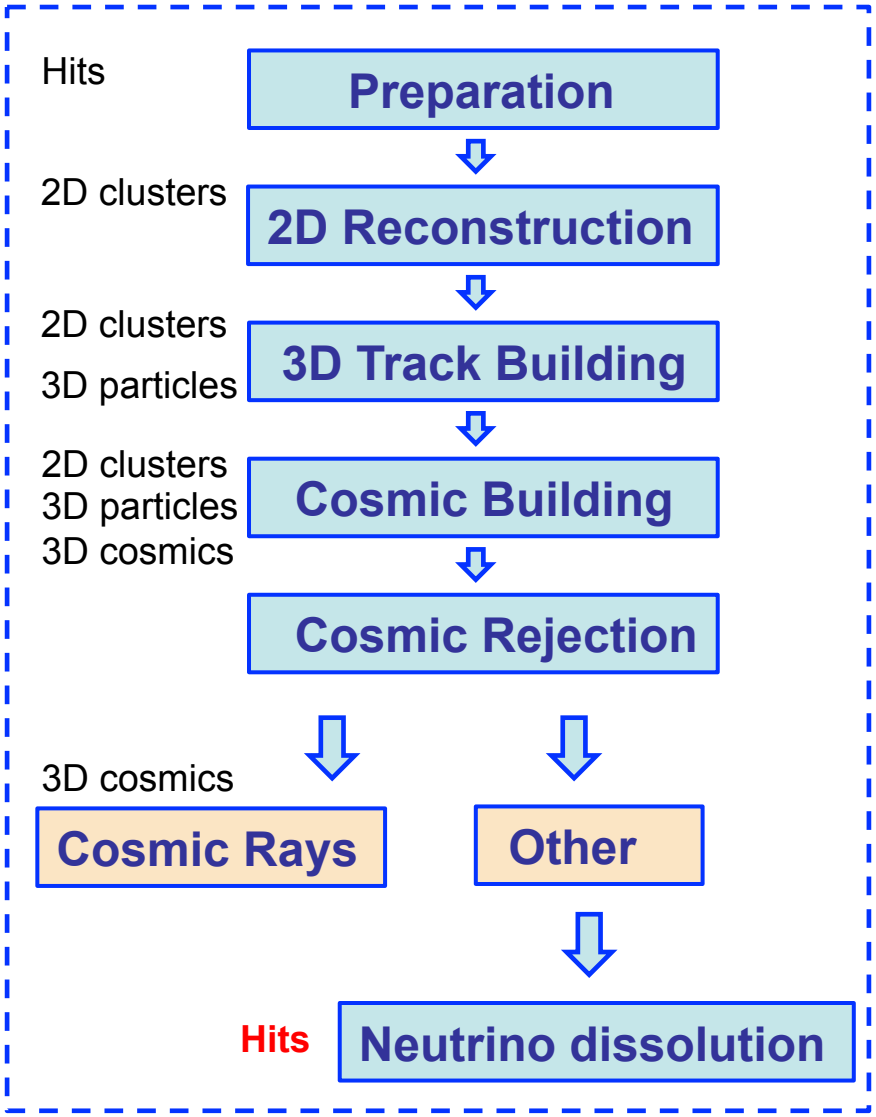
To 3D or not to 3D

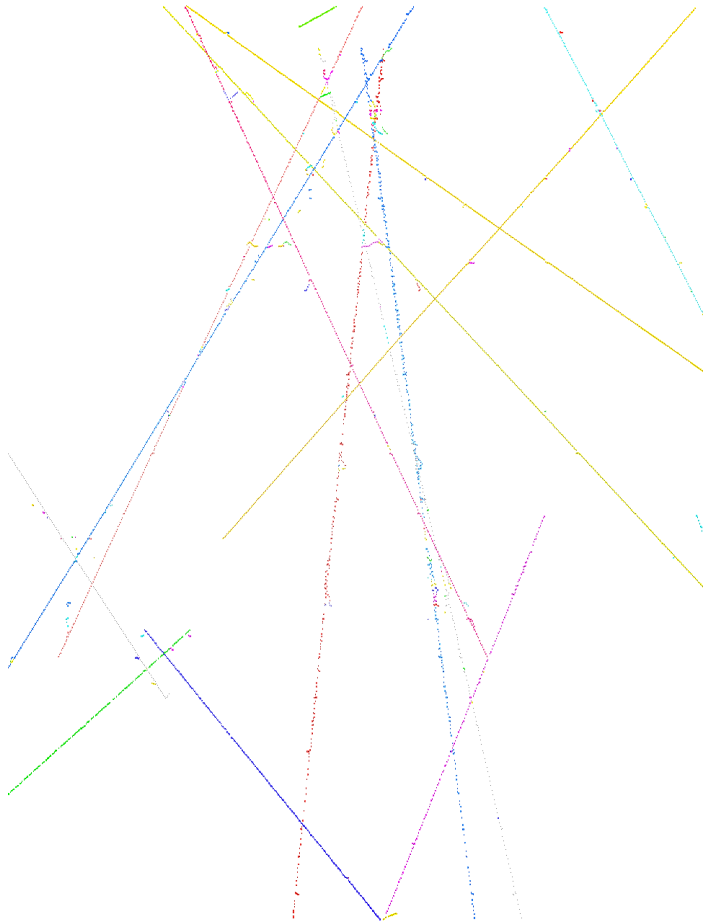
- ★ **Major question: when to go 3D ?**
- ★ **Possibilities**
  - **As early as possible**
    - **PRO:** this is what we would like to do
    - **CON:** there will be ambiguities, **will make mistakes**
  - **As late as possible**
    - **PRO:** we are dealing with inherently 2D readout. x vs time
    - **CON:** in some views cannot resolve particles, **will make mistakes**



- ★ **Or better still… prevaricate**
  - **Defer the decision**
    - Be willing to live with 2D and 3D clusters
    - Gradually move to 3D
    - Try to avoid mistakes

MicroBooNE Strategy





Cosmic path

Hits

Preparation

2D clusters

2D Reconstruction

2D clusters
3D particles

3D Track Building

2D clusters
3D particles
3D cosmics

Cosmic Building

Cosmic Rejection

3D cosmics

Cosmic Rays

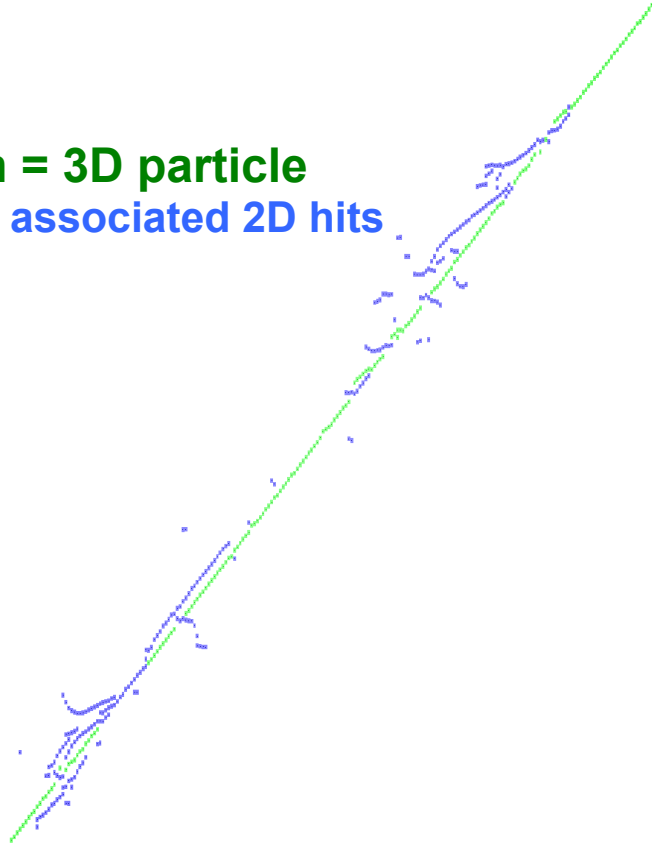
Other

Hits

Neutrino dissolution

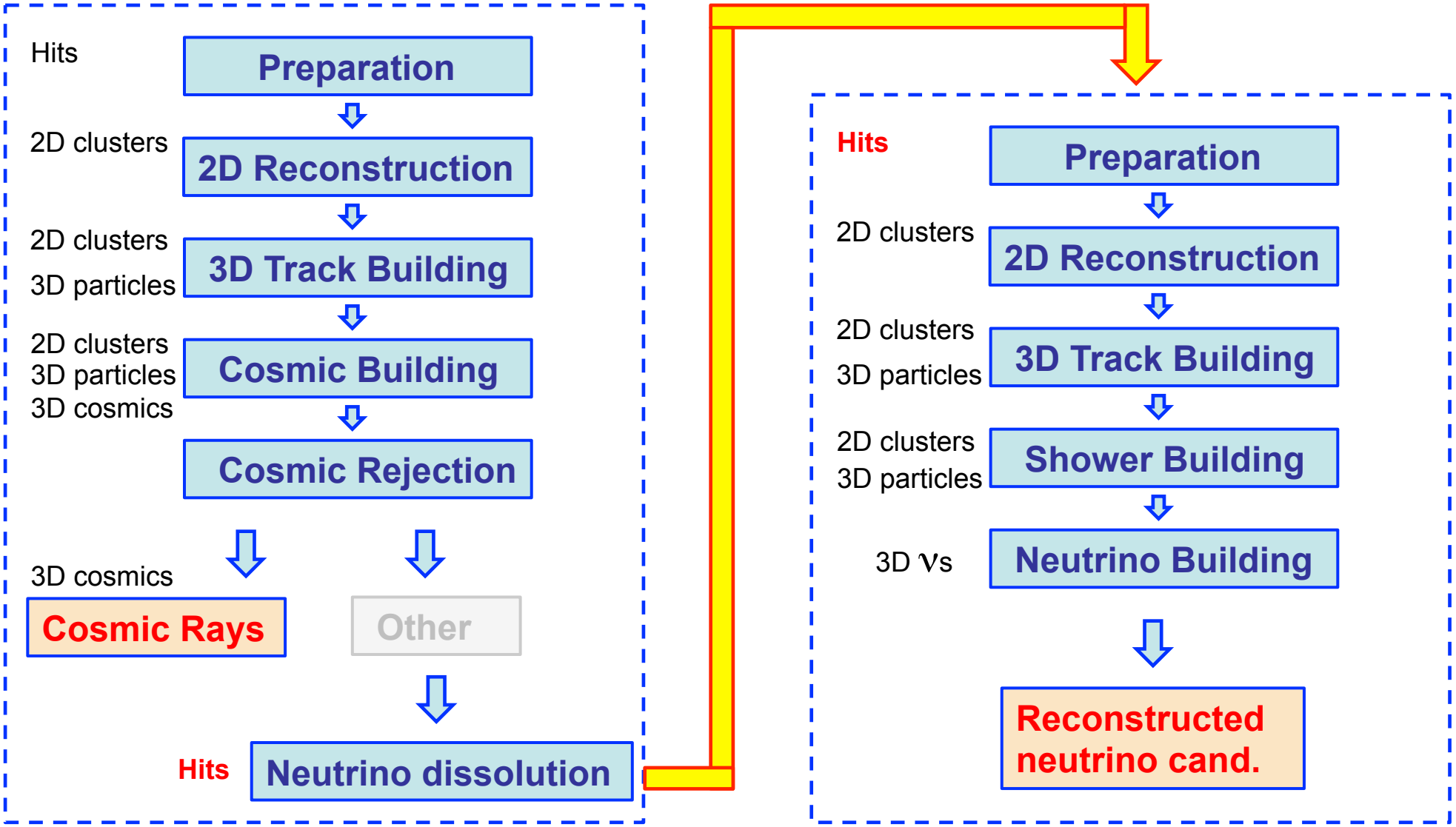
Dressed **Cosmic Rays**

Green = 3D particle
Blue = associated 2D hits

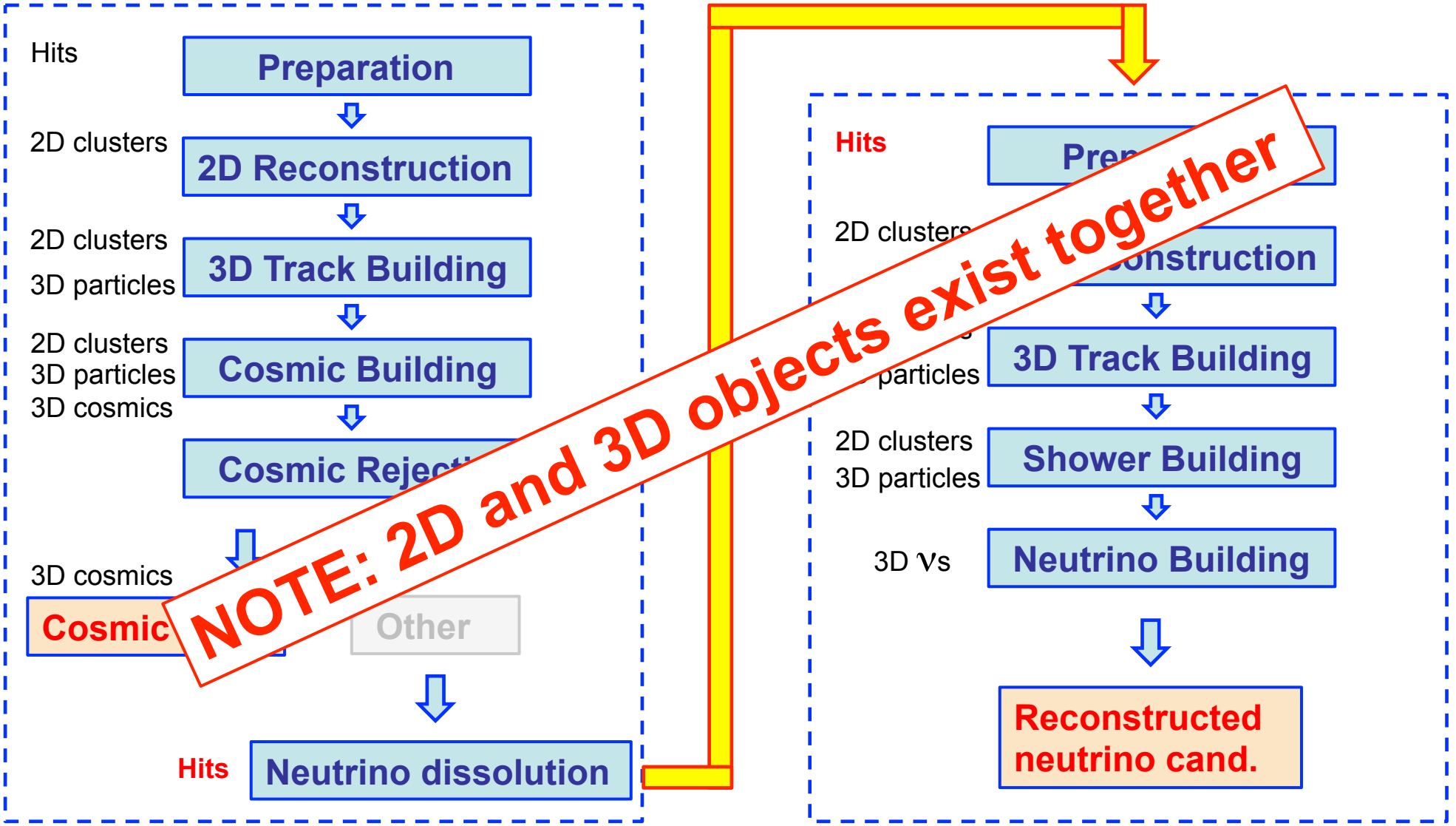


Candidate **Neutrino Hits**

Full path


Hits
Preparation
2D clusters
2D Reconstruction
2D clusters
3D particles
3D Track Building
2D clusters
3D particles
3D cosmics
Cosmic Building
Cosmic Rejection
3D cosmics
Cosmic Rays
Other
Hits
Neutrino dissolution
Hits
Preparation
2D clusters
2D Reconstruction
2D clusters
3D particles
3D Track Building
2D clusters
3D particles
Shower Building
3D νs
Neutrino Building
Reconstructed neutrino cand.

Full path

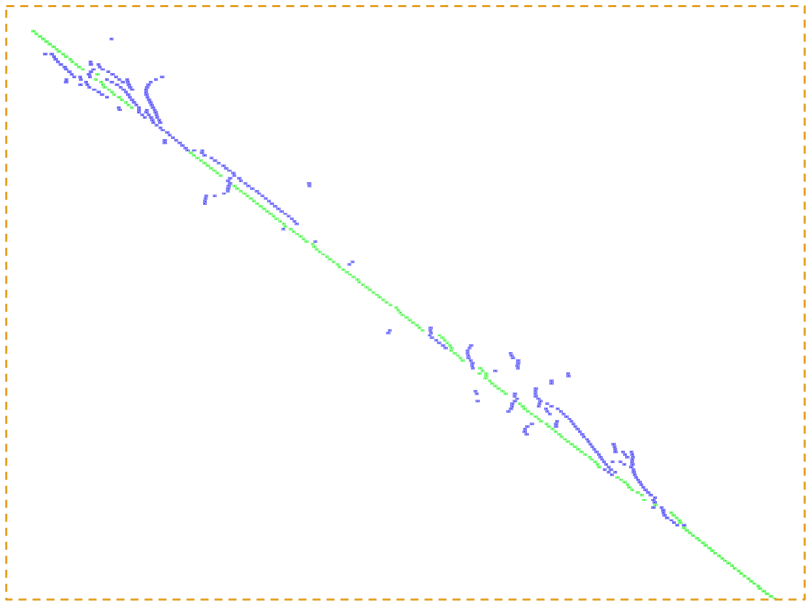

Hits
Preparation
2D clusters
2D Reconstruction
2D clusters
3D particles
3D Track Building
2D clusters
3D particles
3D cosmics
Cosmic Building
Cosmic Reject
3D cosmics
Cosmic
Other
Hits
Neutrino dissolution
NOTE: 2D and 3D objects exist together
Hits
Prep
2D clusters
onstruction
particles
3D Track Building
2D clusters
3D particles
Shower Building
3D νs
Neutrino Building
Reconstructed neutrino cand.

Status

★ **We have now implemented this full reconstruction chain in the sense that the series of Algorithms exists:**

- **Some Algorithms are still placeholders**
- **Cosmic-ray identification (as opposed to reco) – cheats (using MC)**
- **3D track finding now implemented**
- **Working on 3D shower reconstruction**

★ **For today – focus on EDM and relation to LArSoft**



Pandora APIs

- **The role of the Client App is to perform the following operations during initialisation:**
  - Create a Pandora instance that will persist until all event processing is complete.
  - Register factories for all the Algorithms that Pandora requires in order to perform the particle flow reconstruction.
  - Ask Pandora to parse a provided PandoraSettings xml file. Pandora will create instances of all the required Algorithms and it will configure them as instructed.
- **On a per-event basis, the following operations are required:**
  - Ask Pandora to create self-describing reconstruction objects representing the event e.g. calorimeter hits and, if desired, MC particles and their associations.
  - Ask Pandora to process these input objects using the provided Algorithm configuration.
  - Ask Pandora to provide the final list of reconstructed particles, so that they may be stored in the Client Application's native framework.
  - Ask Pandora to reset everything for future event processing.

Pandora Algs

- The Pandora reconstruction philosophy is to use a large number of Algorithms, each of which aims to carefully address a specific event topology. It is crucial to avoid making mistakes.
- The Pandora Algorithm interface class is rather simple, containing purely virtual ReadSettings, Initialize and Run methods (receive callbacks), plus default Constructor and virtual Destructor.
- Importantly, Algorithms can use the PandoraContentAPIs (discussed later) to access the Pandora reconstruction objects and to perform non-const operations on these objects.

- Users create derived Algorithm classes, which access the Pandora objects and perform specific pattern-recognition tasks, typically creating/merging/splitting Clusters or Particles.
- Memory management for the Pandora objects is performed by the Pandora framework, so the Algorithm implementation should aim to be physics-driven and simple to read/understand.
- Configurable parameters can be specified when the Algorithm receives its ReadSettings callback. Parameters can be mandatory, or can have default values that can be overridden.
- Algorithms must register a Factory class, which allows Pandora to create instances of the derived Alg when required (framework then uses only pointers to the base class).

Pandora EDM

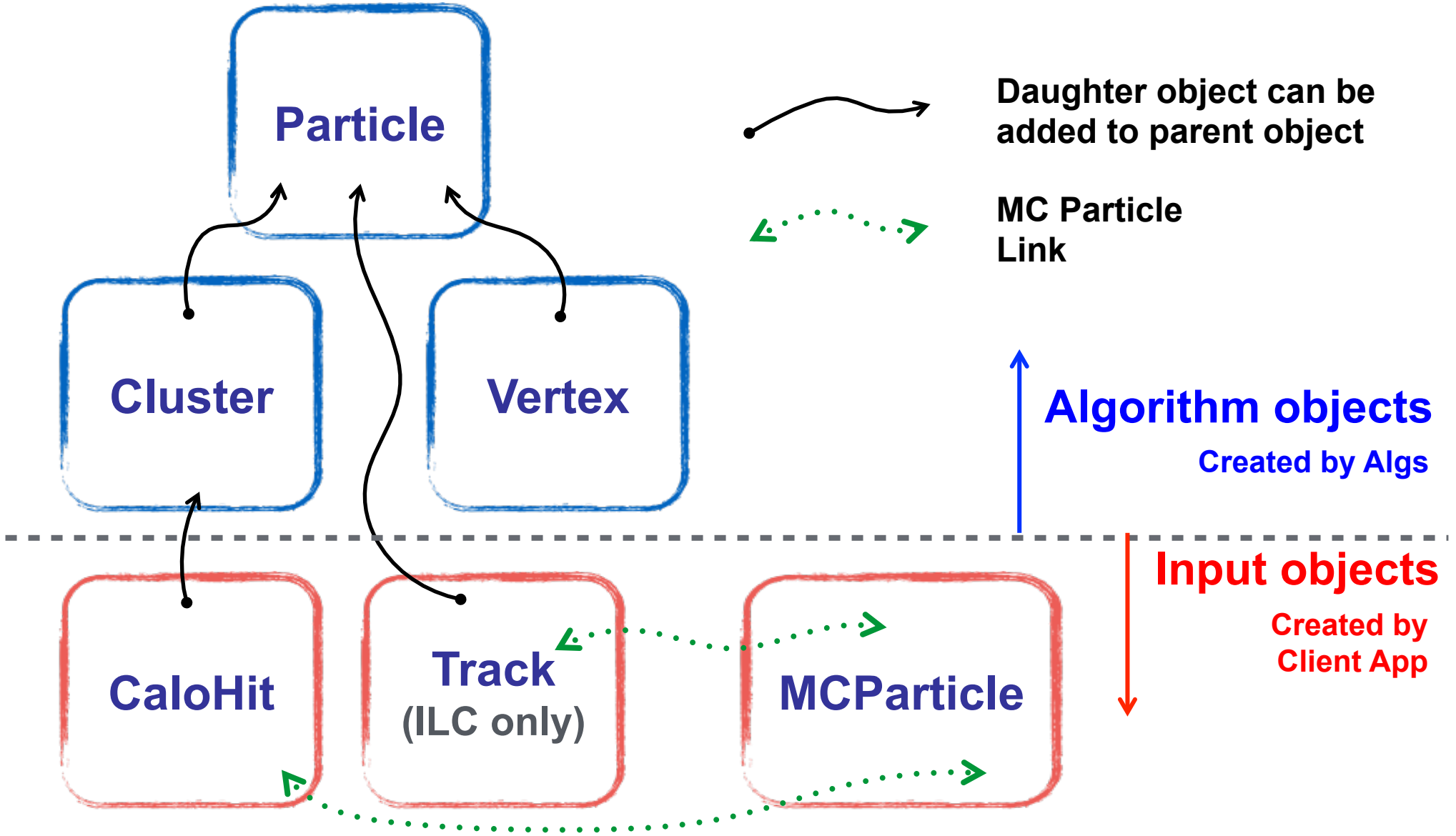
★ **Event Data Model (EDM)**
- Particles
- Clusters
- Tracks
- Calorimeter Hits
- MC Particles
- "Vertices"

Designed for ILC

★ **LAr Interpretation**
- Particles
- Clusters
- [Tracks]
- Calorimeter Hits
- MC Particles

Pandora returns high-level particles

Pandora Event Data Model



e.g. Pandora Cluster

```
const OrderedCaloHitList &GetOrderedCaloHitList() const;
const CaloHitList &GetIsolatedCaloHitList() const;
unsigned int GetNCaloHits() const;
unsigned int GetNIsolatedCaloHits() const;
unsigned int GetNPossibleMipHits() const;
float GetMipFraction() const;
float GetElectromagneticEnergy() const;
float GetHadronicEnergy() const;
float GetIsolatedElectromagneticEnergy() const;
float GetIsolatedHadronicEnergy() const;
bool IsFixedPhoton() const;
bool IsFixedElectron() const;
bool IsFixedMuon() const;
bool IsMipTrack() const;
bool IsTrackSeeded() const;
const Track *GetTrackSeed() const;
PseudoLayer GetInnerPseudoLayer() const;
PseudoLayer GetOuterPseudoLayer() const;
bool ContainsHitInOuterSamplingLayer() const;
bool ContainsHitType(const HitType hitType) const;
const CartesianVector GetCentroid(const PseudoLayer pseudoLayer) const;
const CartesianVector &GetInitialDirection() const;
const ClusterHelper::ClusterFitResult &GetFitToAllHitsResult() const;
float GetCorrectedElectromagneticEnergy() const;
float GetCorrectedHadronicEnergy() const;
float GetTrackComparisonEnergy() const;
PseudoLayer GetShowerStartLayer() const;
float GetShowerProfileStart() const;
float GetShowerProfileDiscrepancy() const;
HitType GetInnerLayerHitType() const;
HitType GetOuterLayerHitType() const;
const TrackList &GetAssociatedTrackList() const;


void SetIsFixedPhotonFlag(bool isFixedPhotonFlag);
void SetIsFixedElectronFlag(bool isFixedElectronFlag);
void SetIsFixedMuonFlag(bool isFixedMuonFlag);
void SetIsMipTrackFlag(bool isMipTrackFlag);
```

- Pandora Clusters are essentially just containers of Pandora CaloHits.
- Hits are ordered by PseudoLayer, (binned wire coordinate for LAr TPC).
- Majority of Pandora Algs will work with Clusters e.g trying to refine them by splitting, merging, etc.

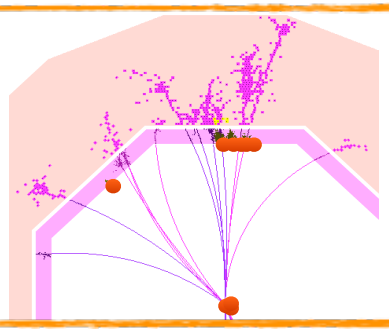
e.g. Pandora Vertex and MCParticle

Vertices currently allow algorithms to define and persist (lists of) space points. Vertices can be added to Particles.

```
const CartesianVector &GetPosition() const;
```

Additional member variables, such as VertexType, may follow if required.

- **Particle vertices** →



MCParticles allow "truth" information to be passed into Pandora in a controlled manner. They can be associated with CaloHits, allowing aspects of the pattern-recognition to be cheated (for debugging)

```
bool IsRootParticle() const;
bool IsPfoTarget() const;
float GetEnergy() const;
const CartesianVector &GetMomentum() const;
const CartesianVector &GetVertex() const;
const CartesianVector &GetEndpoint() const;
float GetInnerRadius() const;
float GetOuterRadius() const;
int GetParticleId() const;
MCParticleType GetMCParticleType() const;
bool IsPfoTargetSet() const;
const MCParticle *GetPfoTarget() const;
Uid GetUid() const;
const MCParticleList &GetParentList() const;
const MCParticleList &GetDaughterList() const;
```

Pandora ParticleFlowObject

★ Output of reconstruction is a list of particles…

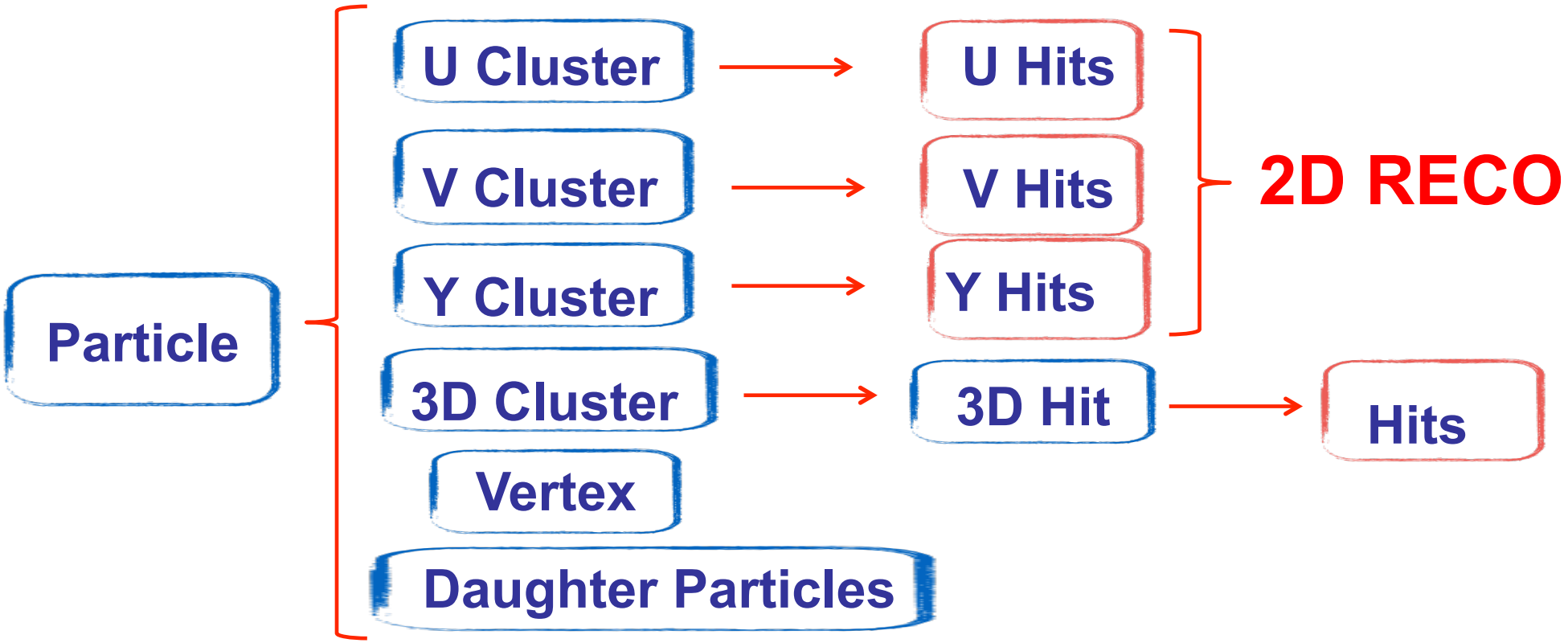
```
int GetParticleId() const;
int GetCharge() const;
float GetMass() const;
float GetEnergy() const;
const CartesianVector &GetMomentum() const;
const TrackList &GetTrackList() const;
const ClusterList &GetClusterList() const;
const VertexList &GetVertexList() const;
TrackAddressList GetTrackAddressList() const;
ClusterAddressList GetClusterAddressList() const;
unsigned int GetNTracks() const;
unsigned int GetNClusters() const;
const PfoList &GetParentPfoList() const;
const PfoList &GetDaughterPfoList() const;
unsigned int GetNParentPfos() const;
unsigned int GetNDaughterPfos() const;


void SetParticleId(const int particleId);
void SetCharge(const int charge);
void SetMass(const float mass);
void SetEnergy(const float energy);
void SetMomentum(const CartesianVector &momentum);
```

Lists of clusters, tracks and vertices

Relation to other particles

Pandora LAr EDM



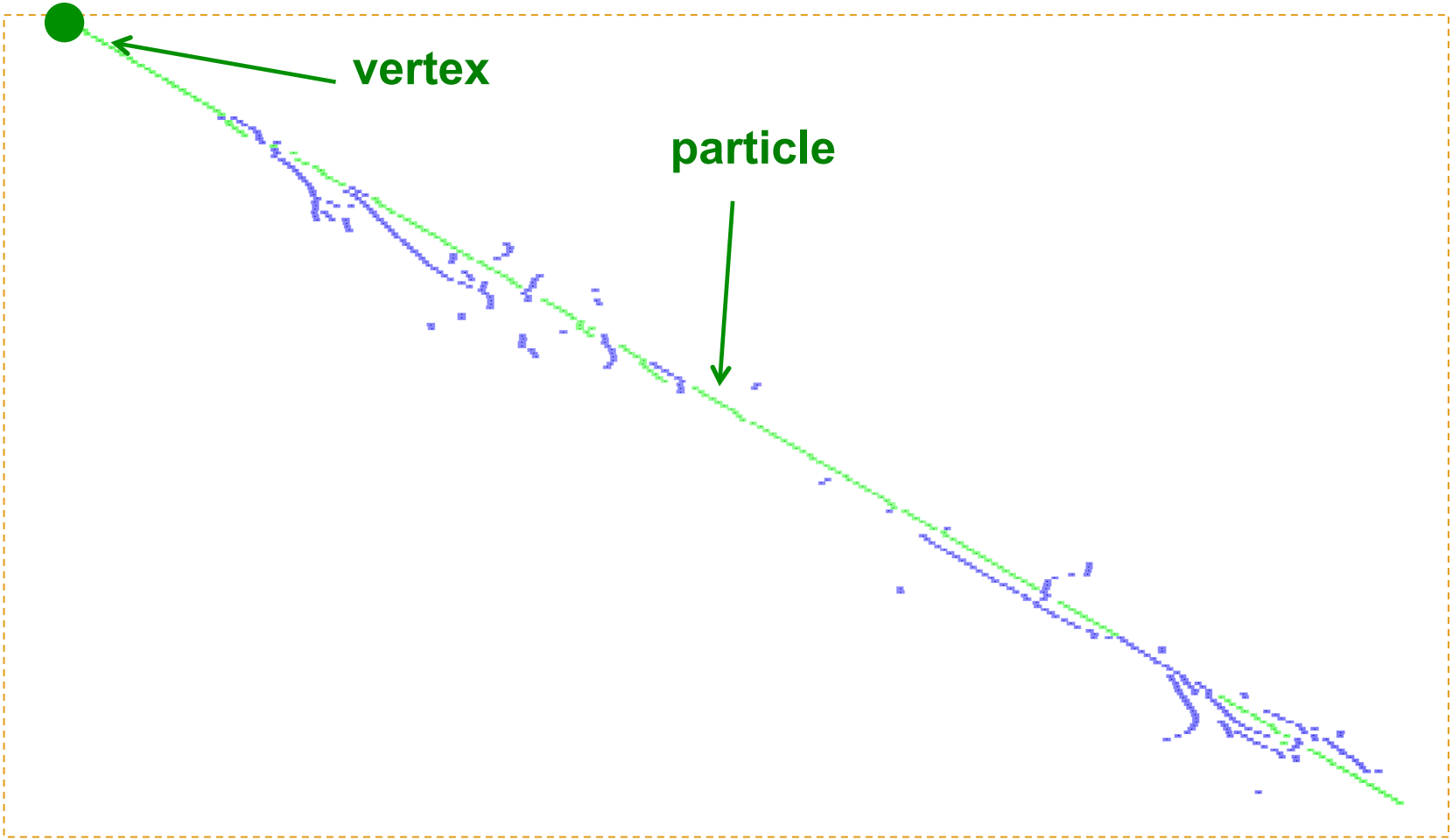
★ **3D Hit:** Ideally, one 3D hit "space point" for each U, V, W hit

★ **Vertex:** 3D "space point" for start coordinate of particle

★ **Daughters:** all particles descending from this particle

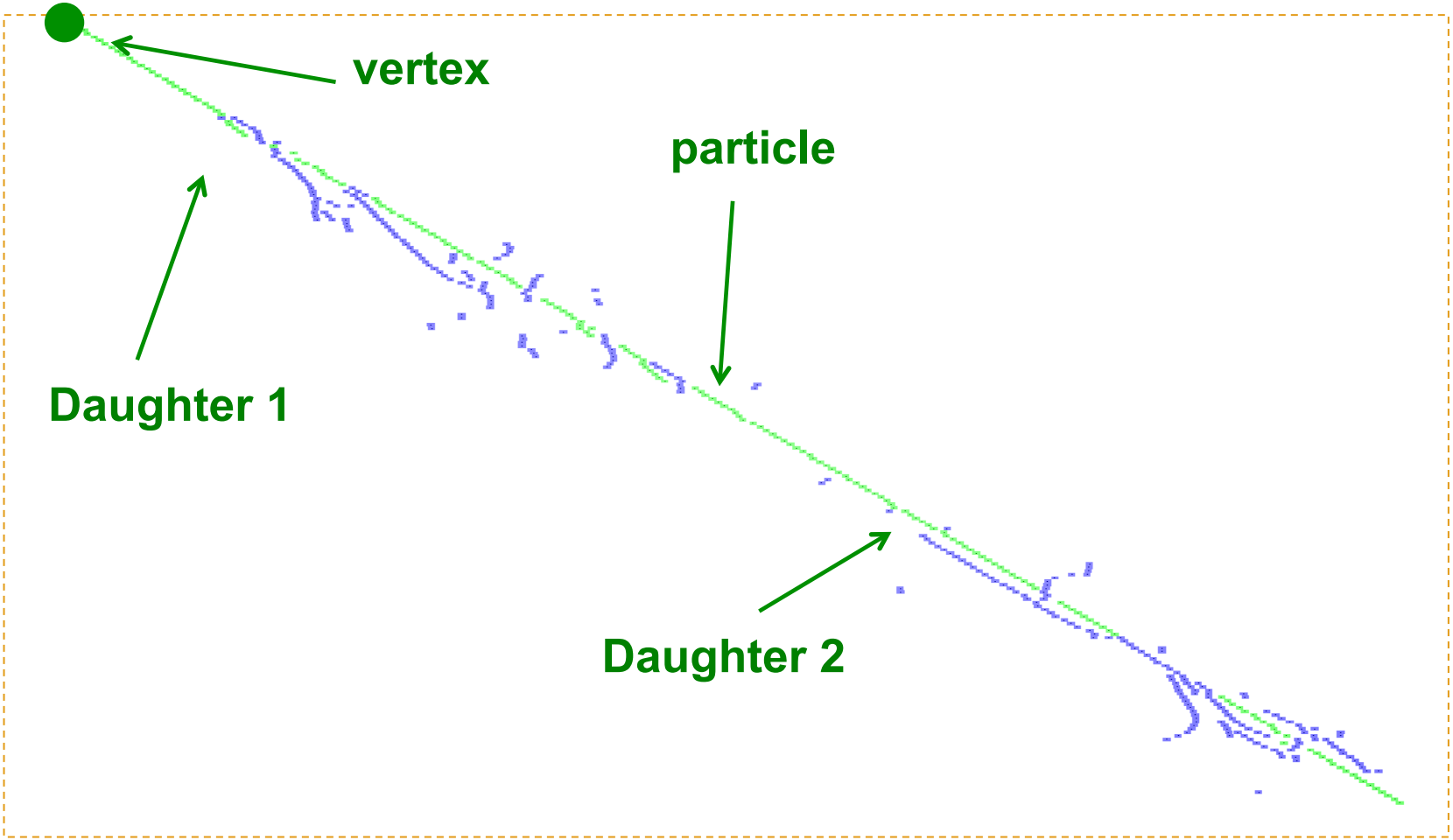
e.g. “a cosmic muon”

★ Primary Particle: the muon “track”



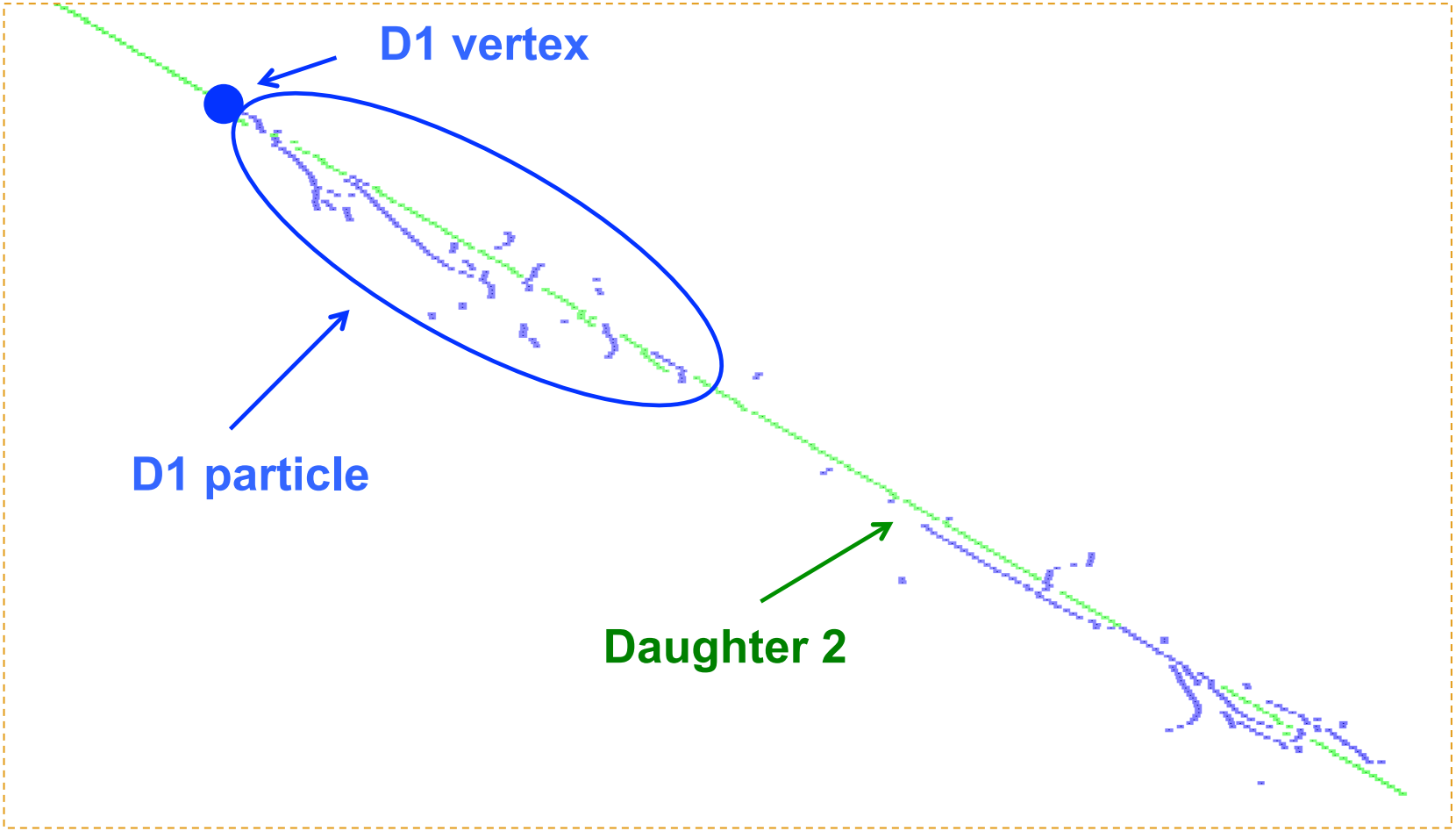
e.g. “a cosmic muon”

★ Primary Particle: the muon “track”



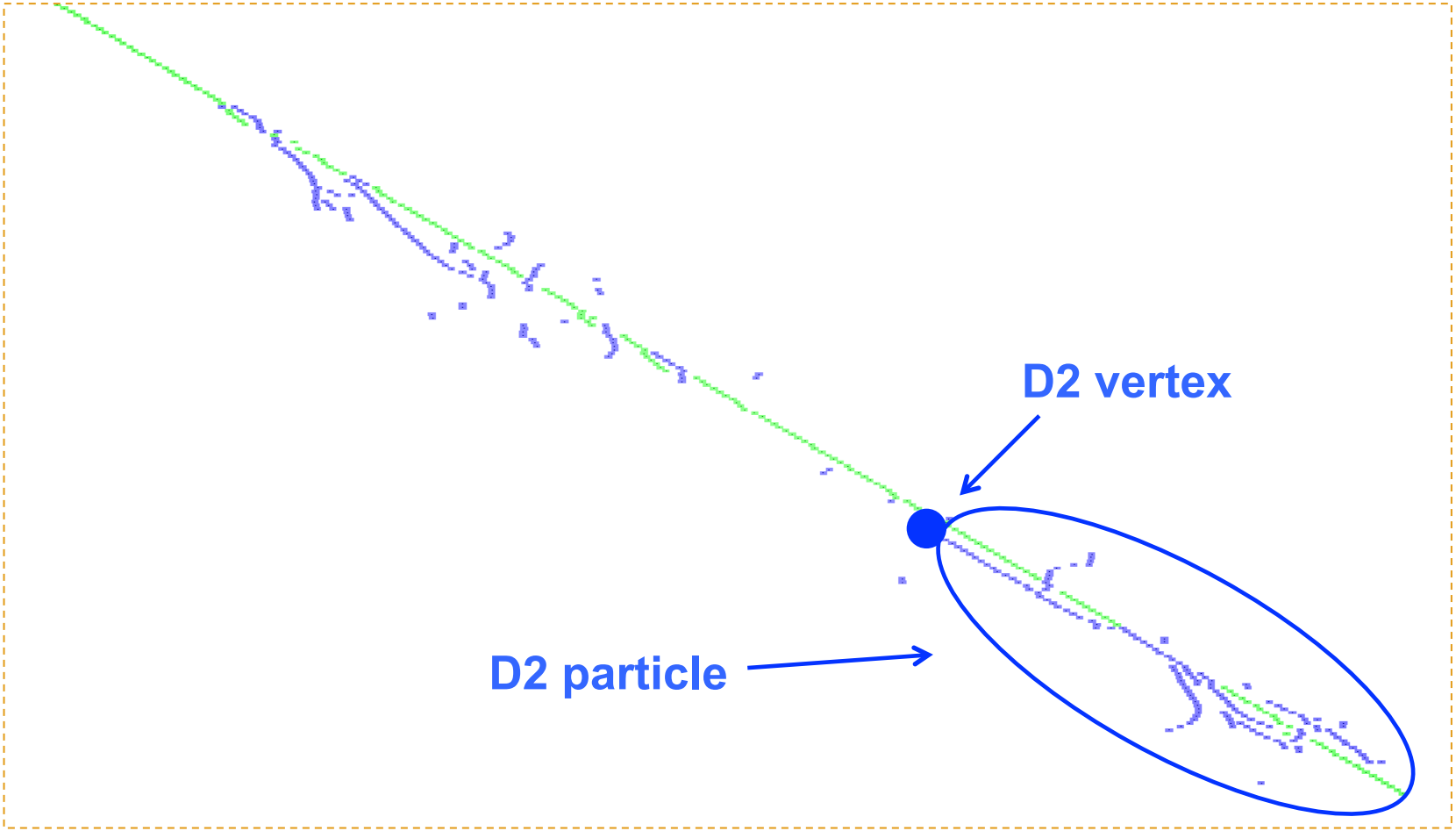
Daughter particles

★ Daughter 1: an EM shower (still a particle)



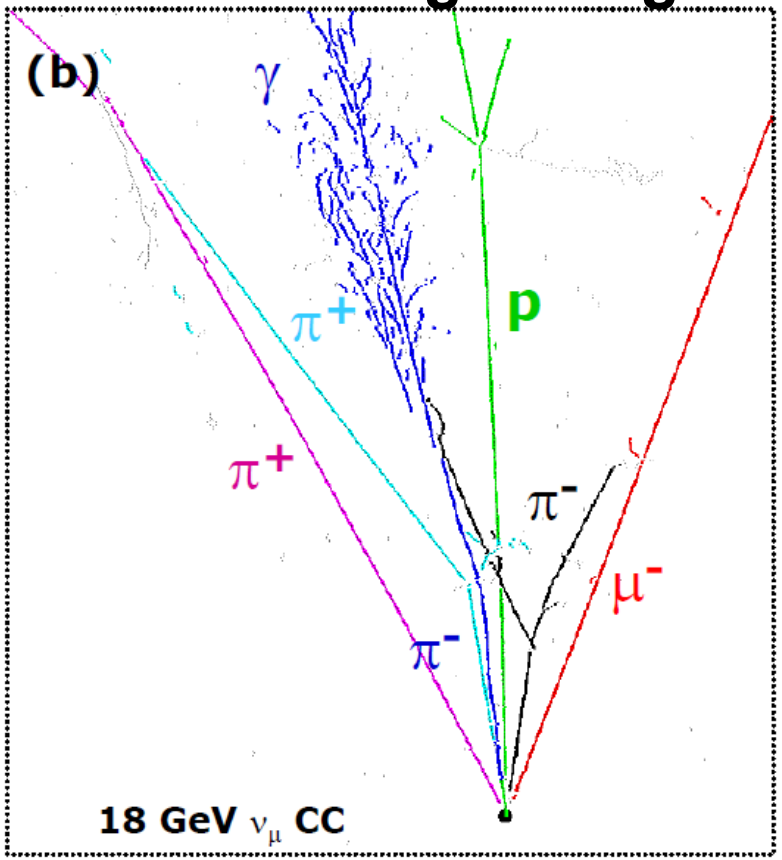
etc.

★ Daughter 2: an EM shower (still a particle)



In an ideal world….

★ **A neutrino = a hierarchy of particles from the primary vtx…**
this is out long-term goal (inside Pandora)



★ **How to persist this information in LArSoft ?**

Summary

- ★ Making rapid progress with algorithms
  - Decent 2D track reconstruction (clusters)
  - Powerful 3D associations and space-point creation
  - Working on 2D shower reconstruction (clusters)
- ★ Aim of reconstruction is hierarchy of particles
- ★ Particles contain
  - Vertex (start point)
  - 2D clusters
  - 3D clusters (collections of 3D hits)
  - List of daughter particles
- ★ At the moment, only the LArSoft interface to output clusters exists