PADME SW & ANALYSIS

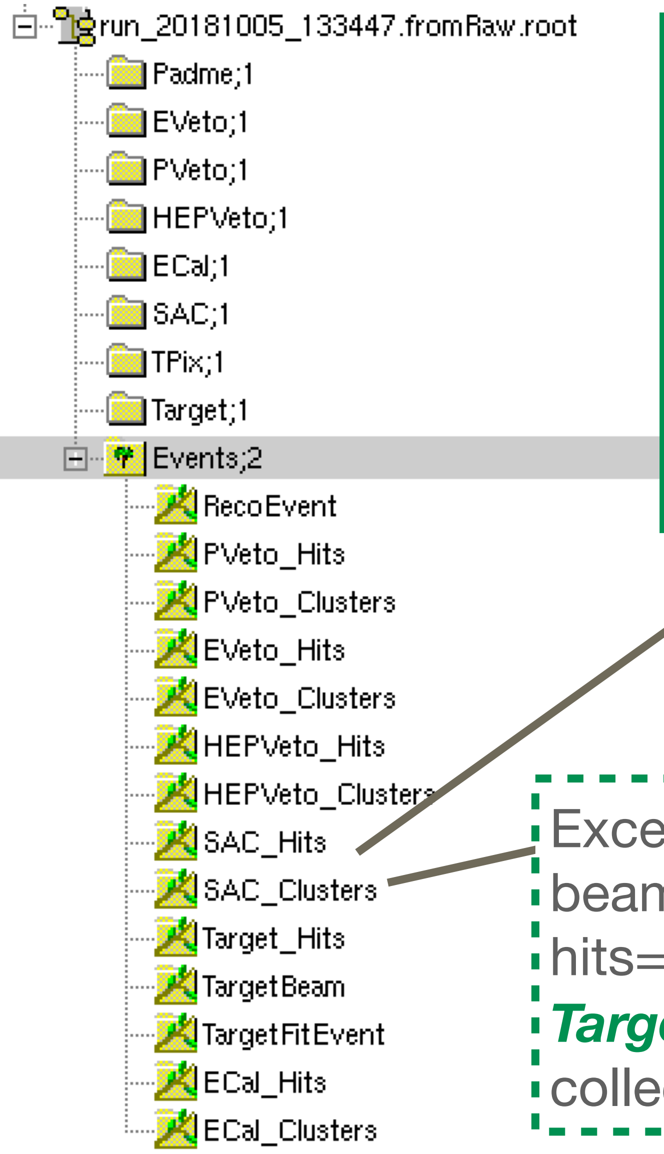
LECCE ACTIVITIES

G. Chiodini, I. Oceano, F. Oliva, V. Scherini, S. Spagnolo

REMINDER

- A version of the reconstruction software (developed in the branch feature/ CusterReco2) ***merged to develop***, implements ***two new features:***
  - ***1)*** Since all reconstruction algorithms must build ***Clusters from Hits***, in the second reconstruction step
    - a common solution to ***"build clusters"*** is implemented in the reco base class and ***common classes for Clusters and Cluster Collections*** are available and centrally made persistent;
  - ***2)*** PadmeReco the **reconstruction** main **can read**
    - *RawEvents* - (input: waveforms in root format, reco. consists of 2 steps: *raw_to_recoHits* and *recoHits_to_Clusters*)
    - *Events* - (input recoHits; reco consists of one step: *recoHits_to_Clusters*)
    - *MCEvent* (input: MC hit/digit collections) - status to be revised

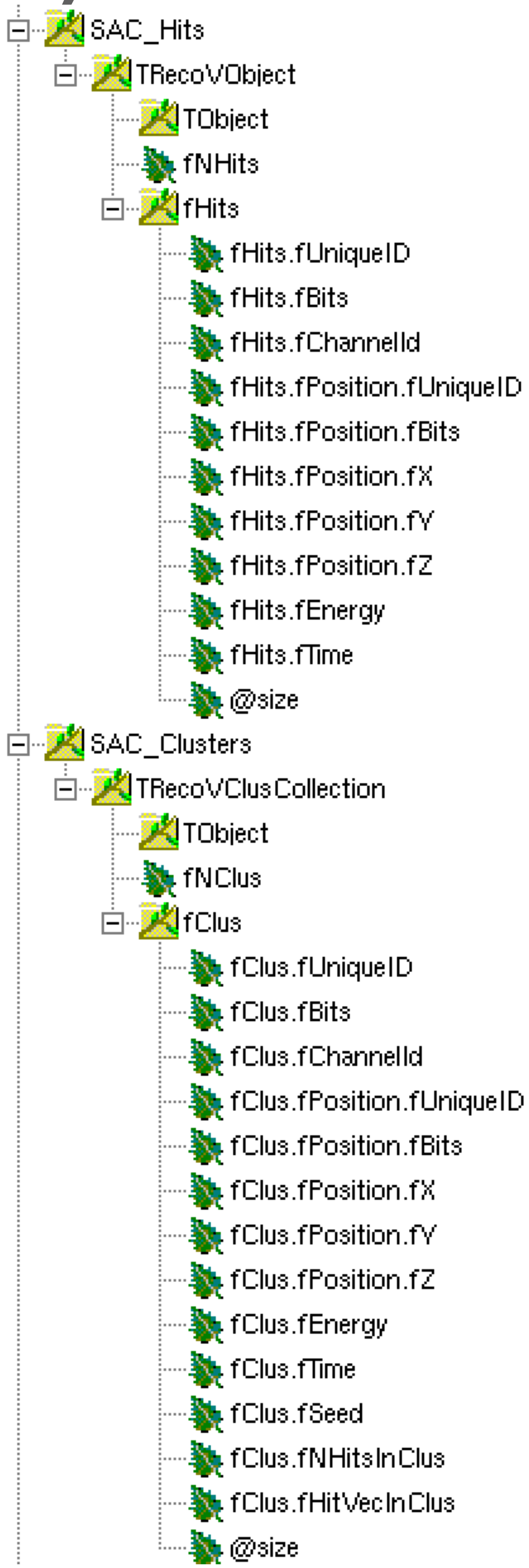
ROOT FILE (OUTPUT OF THE RECO)



```
run_20181005_133447.fromRaw.root
    Padme;1
    EVeto;1
    PVeto;1
    HEPVeto;1
    ECal;1
    SAC;1
    TPix;1
    Target;1
    Events;2
        RecoEvent
        PVeto_Hits
        PVeto_Clusters
        EVeto_Hits
        EVeto_Clusters
        HEPVeto_Hits
        HEPVeto_Clusters
        SAC_Hits
        SAC_Clusters
        Target_Hits
        TargetBeam
        TargetFitEvent
        ECal_Hits
        ECal_Clusters
```

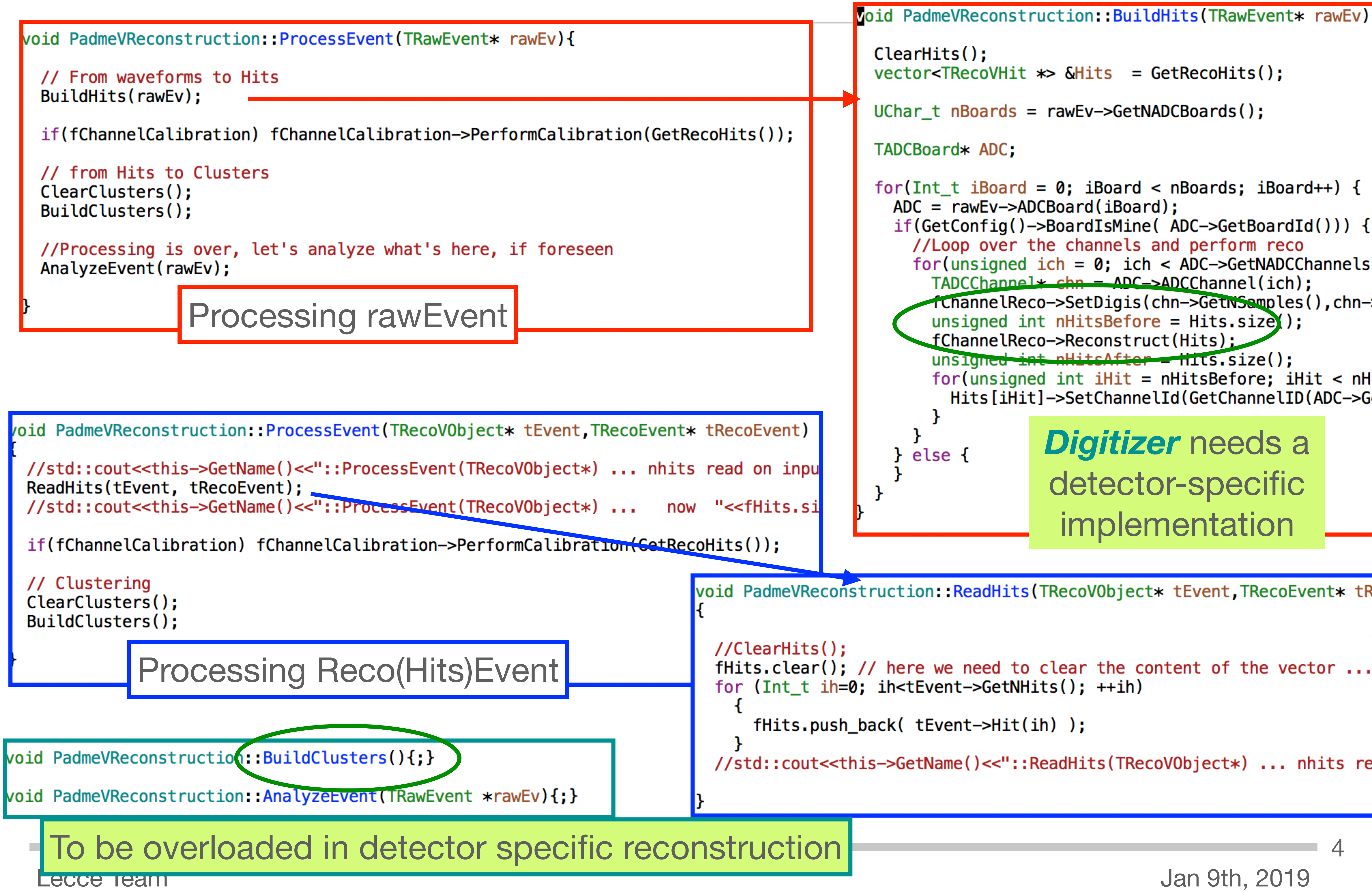
- In the tree
  - One branch per detector / Hits
  - One branch per detector / Clusters

Exception: ***Target*** that builds the beam profile in x and y from hits=charge on each strip; ***TargetBeam*** instead of cluster collection



```
SAC_Hits
    TRecoVObject
        TObject
        fNHits
        fHits
            fHits.fUniqueID
            fHits.fBits
            fHits.fChannelId
            fHits.fPosition.fUniqueID
            fHits.fPosition.fBits
            fHits.fPosition.fX
            fHits.fPosition.fY
            fHits.fPosition.fZ
            fHits.fEnergy
            fHits.fTime
            @size
SAC_Clusters
    TRecoVClusCollection
        TObject
        fNClus
        fClus
            fClus.fUniqueID
            fClus.fBits
            fClus.fChannelId
            fClus.fPosition.fUniqueID
            fClus.fPosition.fBits
            fClus.fPosition.fX
            fClus.fPosition.fY
            fClus.fPosition.fZ
            fClus.fEnergy
            fClus.fTime
            fClus.fSeed
            fClus.fNHitsInClus
            fClus.fHitVecInClus
            @size
```

RECONSTRUCTION FROM RAW OR RECOHITS



```
void PadmeVReconstruction::ProcessEvent(TRawEvent* rawEv){

  // From waveforms to Hits
  BuildHits(rawEv);

  if(fChannelCalibration) fChannelCalibration->PerformCalibration(GetRecoHits());

  // from Hits to Clusters
  ClearClusters();
  BuildClusters();

  //Processing is over, let's analyze what's here, if foreseen
  AnalyzeEvent(rawEv);

}
```

Processing rawEvent

```
void PadmeVReconstruction::BuildHits(TRawEvent* rawEv)

  ClearHits();
  vector<TRecoVHit *> &Hits  = GetRecoHits();

  UChar_t nBoards = rawEv->GetNADCBoards();

  TADCBoard* ADC;

  for(Int_t iBoard = 0; iBoard < nBoards; iBoard++) {
    ADC = rawEv->ADCBoard(iBoard);
    if(GetConfig()->BoardIsMine( ADC->GetBoardId())) {
      //Loop over the channels and perform reco
      for(unsigned ich = 0; ich < ADC->GetNADCChannels
        TADCChannel* chn = ADC->ADCChannel(ich);
        fChannelReco->SetDigis(chn->GetNSamples(),chn-
        unsigned int nHitsBefore = Hits.size();
        fChannelReco->Reconstruct(Hits);
        unsigned int nHitsAfter = Hits.size();
        for(unsigned int iHit = nHitsBefore; iHit < nH
          Hits[iHit]->SetChannelId(GetChannelID(ADC->G
        }
      }
    } else {
    }
  }
}
```

Digitizer needs a detector-specific implementation

```
void PadmeVReconstruction::ProcessEvent(TRecoVObject* tEvent,TRecoEvent* tRecoEvent)
{
  //std::cout<<this->GetName()<<"::ProcessEvent(TRecoVObject*) ... nhits read on inpu
  ReadHits(tEvent, tRecoEvent);
  //std::cout<<this->GetName()<<"::ProcessEvent(TRecoVObject*) ...   now  "<<fHits.si

  if(fChannelCalibration) fChannelCalibration->PerformCalibration(GetRecoHits());

  // Clustering
  ClearClusters();
  BuildClusters();

}
```

Processing Reco(Hits)Event

```
void PadmeVReconstruction::ReadHits(TRecoVObject* tEvent,TRecoEvent* tR
{

  //ClearHits();
  fHits.clear(); // here we need to clear the content of the vector ...
  for (Int_t ih=0; ih<tEvent->GetNHits(); ++ih)
    {
      fHits.push_back( tEvent->Hit(ih) );
    }
  //std::cout<<this->GetName()<<"::ReadHits(TRecoVObject*) ... nhits re

}
```

```
void PadmeVReconstruction::BuildClusters(){;}

void PadmeVReconstruction::AnalyzeEvent(TRawEvent *rawEv){;}
```

To be overloaded in detector specific reconstruction

DETECTOR SPECIFIC CODE

- How to write Hit/Cluster Reconstruction for a generic PADME Detector:
  - very few blocks of code to be implemented:

  - HITs:
    - implement DigitizerChannelXXX (in PadmeReco/RecoBase)
  - Clusters
    - Implement XXXReconstruction::BuildClusters()

DATA PROCESSING EXERCISE - A FEW FIGURES

Processing rawEvent

- Run ***run_0000000_20181217 about 2.5M bunches*** copied in Lecce (raw) and processed on slc6
- Raw: ***Size on disk 1.72 TB***
  - processed with padme-fw develop branch: ***output size 27.3 GB***
  - average nPOT/bunch ***25960***
  - total nPOT=***6.4493e+10*** no quality cuts applied
  - reco job (for each lvl1 stream-> 5 jobs) organised as follows:
    - for each run
      - raw file copied locally from Lecce nfs storage
      - run PadmeReco (*) ***real 1m44.476s***
      - output copied to Lecce nfs storage
    - executed on slc6 (Lecce nodes)
    - Output ready after <15h for 5 jobs lunched in parallel

(*) from local installation of padme-fw / develop branch

DATA PROCESSING EXERCISE - A FEW FIGURES

Processing rawEvent

- Run ***run_0000000_20181217 about 2.5M bunches*** copied in Lecce (raw) and processed on slc6
  - RawToRecoHit jobs (processing 10k events in a single job)
    - ** PadmeReco MAIN *after recoIO init.* SZ= 143 Mb Time = 0.17 s DeltaM = 2.8 Mb Delta T =0.05 s
    - ** PadmeReco MAIN *after Reconstruction init.* SZ= 149 Mb Time = 0.33 s DeltaM = 6.2 Mb Delta T =0.16 s
    - ** PadmeReco MAIN *after first event* SZ= 196.5 Mb Time = 0.48 s DeltaM = 47.7 Mb Delta T =0.22 s (for n=2000 events)
    - ** PadmeReco MAIN *after event loop* SZ= 236 Mb Time = 2089 s DeltaM in the loop = 396.4 Mb Delta T =209 s ***Events processed = 2000***
      - **AVERAGE mem leak/event SZ= 19.8 Kb/event;** ***average*** **total*_cpuTime/event = 105 ms (does not include initialization)***
      - **Good ! HOWEVER**
        - running on more events …

DATA PROCESSING EXERCISE - A FEW FIGURES

Processing rawEvent

- from the log

standard

```
=== Read raw event in position 2400 ===
--- PadmeReconstruction --- run/event/time 0 17000 2018-12-17 20:47:34.208654198Z
***** PadmeReco MAIN after this event          SZ= 244736 Kb   Time = 266.23 seconds  ----     DeltaM = 0       Delta T =0.0899
=== Read raw event in position 2500 ===
--- PadmeReconstruction --- run/event/time 0 17500 2018-12-17 20:47:44.356518707Z
=== Read raw event in position 2600 ===
--- PadmeReconstruction --- run/event/time 0 18000 2018-12-17 20:47:54.518821225Z
=== Read raw event in position 2700 ===
--- PadmeReconstruction --- run/event/time 0 18500 2018-12-17 20:48:04.715157475Z
=== Read raw event in position 2800 ===
--- PadmeReconstruction --- run/event/time 0 19000 2018-12-17 20:48:14.874564406Z
=== Read raw event in position 2900 ===
--- PadmeReconstruction --- run/event/time 0 19500 2018-12-17 20:48:24.996519148Z
Warning in <TSpectrum::SearchHighRes>: Peak buffer full
***** PadmeReco MAIN after this event          SZ= 466136 Kb   Time = 320.59 seconds  ----     DeltaM = 0       Delta T =0.100006
=== Read raw event in position 3000 ===
```

a sudden jump in memory
due to the TSpectrum error ???
to be understood / cured

DATA PROCESSING EXERCISE - A FEW FIGURES

Processing recoHits

- Run ***run_0000000_20181217 size 27.3 GB***
  - ** PadmeReco MAIN *after Reconstruction init.* SZ= 152 Mb Time = 32.3 s
  - ** PadmeReco MAIN *after first event* SZ= 392 Mb Time = 32.46 s why so big ?
    - **AVERAGE mem leak/event SZ= 6 Kb/event;**
      average* total*_cpuTime/event = 2.6 ms (does not include initialization)
    - **HOWEVER:**

```
=== Read (from Hits) event in position 2400 ===
--- PadmeReconstruction --- run/event/time 0 0 1.54593e+09
***** PadmeReco MAIN after this event      SZ= 406952 Kb   Time = 38.86 seconds  ----     DeltaM = 0          Delta T =0
=== Read (from Hits) event in position 2500 ===
--- PadmeReconstruction --- run/event/time 0 0 1.54593e+09
Warning in <Fit>: Fit data is empty
=== Read (from Hits) event in position 2600 ===
--- PadmeReconstruction --- run/event/time 0 0 1.54593e+09
Warning in <Fit>: Fit data is empty
=== Read (from Hits) event in position 2700 ===
--- PadmeReconstruction --- run/event/time 0 0 1.54593e+09
=== Read (from Hits) event in position 2800 ===
--- PadmeReconstruction --- run/event/time 0 0 1.54593e+09
=== Read (from Hits) event in position 2900 ===
--- PadmeReconstruction --- run/event/time 0 0 1.54593e+09
***** PadmeReco MAIN after this event      SZ= 622852 Kb   Time = 40.51 seconds  ----     DeltaM = 0          Delta T =0
```

a sudden jump in memory
to be understood / cured

RECO ON MC

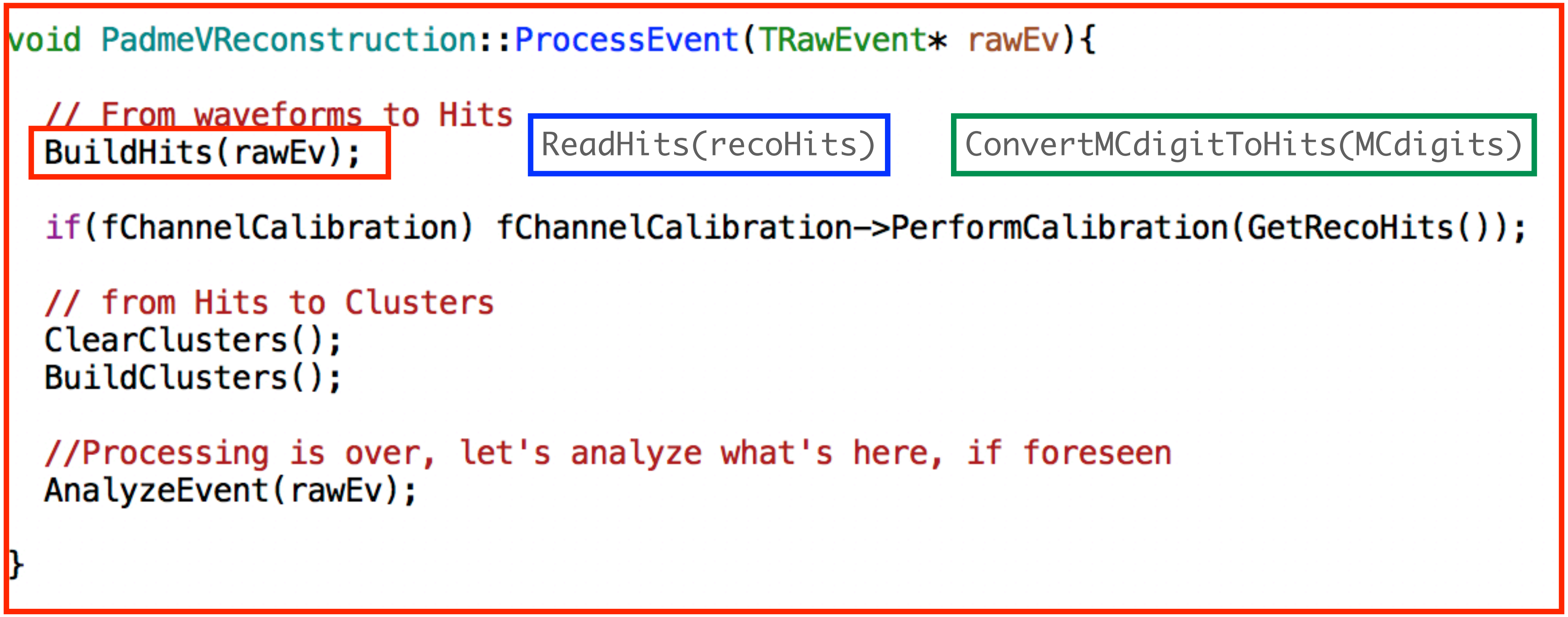
- in develop
- we need to review the status: for each detector
    - what is the input ?
        - hits (G4) or hits+digits
            - what’s the digitization output for each detector ?
    - what is the status of reconstruction:
        - is the reconstruction running on real data able to run on MC ?

RECO ON MC

answers from TARGET

- in develop
- we need to review the status: for each detector
  - what is the input ?
    - hits (G4) or hits+**digits** (*digits* are needed as input), otherwise digitization is run first [unpractical])
      - what's the digitization output for each detector ?
        - ***no Target digitization in PadmeMC in develop*** 🙁
        - detailed work done in the past MCdigits (different class with respect to TMCVhit = RecoHits) but easy to convert 😃
          - [no consistency of position, channel ID between RecoHits and MCdigits …]
        - MCdigits can be produced via *fast digitization [to be the default]* or full digitization, passing via a careful emulation of waveforms 😃
  - what is the status of reconstruction:
    - is the reconstruction running on real data able to run on MC ?
      - ***nothing working at the moment*** 🙁
      - A `MCdigitToRecoHit` - Converter is needed

RECONSTRUCTION FROM RAW OR RECOHITS OR MCDIGITS



```
void PadmeVReconstruction::ProcessEvent(TRawEvent* rawEv){

  // From waveforms to Hits
  BuildHits(rawEv);          ReadHits(recoHits)     ConvertMCdigitToHits(MCdigits)

  if(fChannelCalibration) fChannelCalibration->PerformCalibration(GetRecoHits());

  // from Hits to Clusters
  ClearClusters();
  BuildClusters();

  //Processing is over, let's analyze what's here, if foreseen
  AnalyzeEvent(rawEv);

}
```

- A possibility for reconstruction on MC
  - by using a base class for the MCdigits we can steer the reconstruction of MC events from the base class
  - `ConvertMCdigitToHits` must be implemented/overloaded for each detector reconstruction.

Analysis Framework

- Very basic framework implemented, shared with some of you
  - PadmeAnalysisMain in PadmeReco
  - Analysis folder in PadmeReco containing several classes:
    - ECalAnalysis, SACAnalysis, etc …
- not in the release because it’s not well organised:
  - desired configuration:
    - PadmeAnalysis directory (parallel to PadmeReco, PadmeMC, PadmeRoot, etc) containing:
      - PadmeAnalysisMain
      - ECalAnalysis folder
      - SACAnalysis “ … etc
      - AnalysisTools “

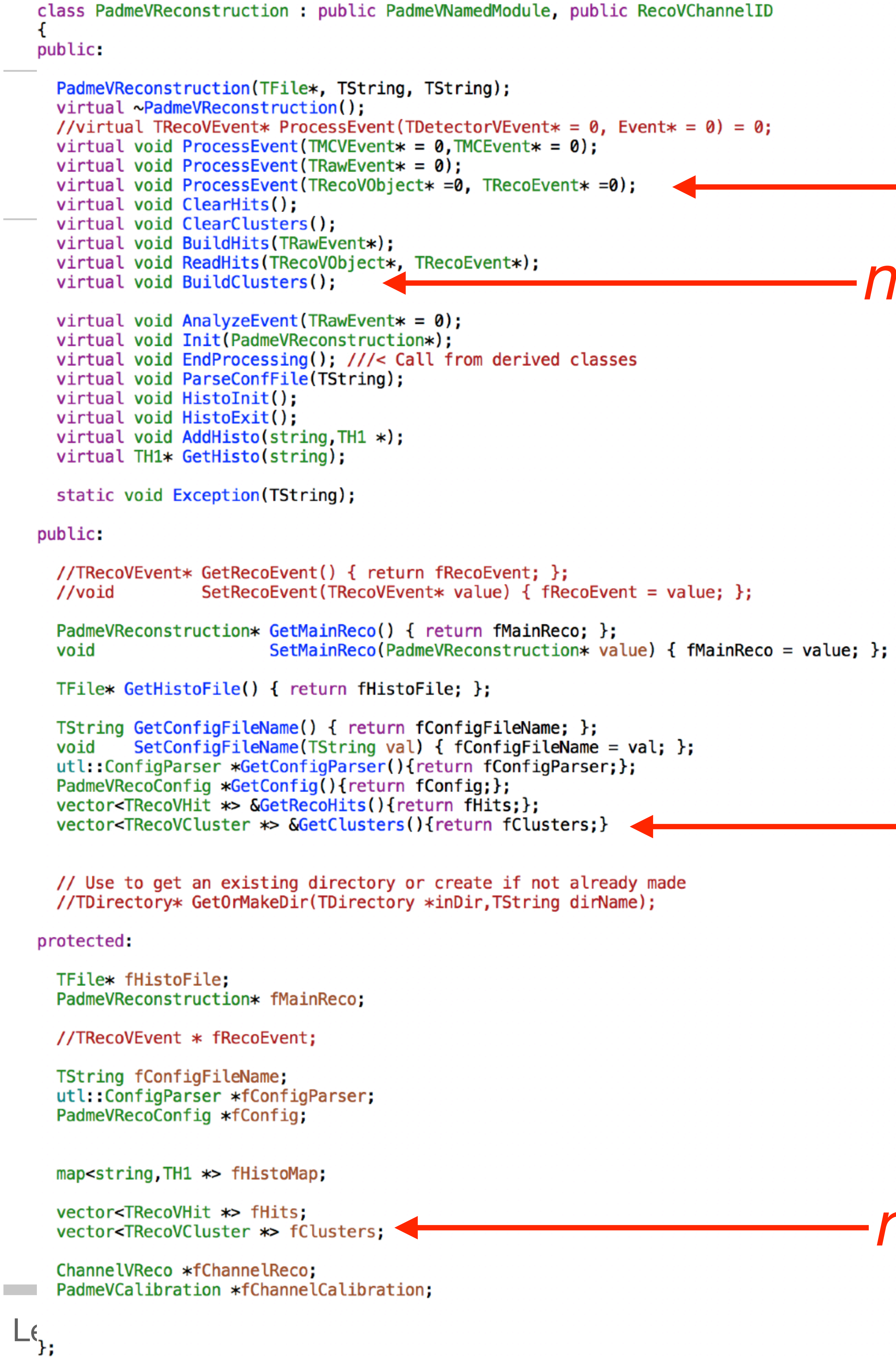
Why ?
Anyone mastering makefiles??

- first task of the PadmeAnalysisMain:
  - run selectorsOfGoodPhysicsObjects (γ, e^+, e^-) for each detector [input: Clusters]
  - requires a minimal/nominal but existing calibration of each detector
  - requires a minimal-global geometry

SUMMARY

- Long TODO list:
  - general tools:
    - triggerTime per board/channel to be integrated (basic algorithm exists)
    - triggerWord: disentangle cosmic from BTF trigger (easy) to be integrated in the fw
  - how far we are from implementing for all detectors [technically, calibration to come afterwards] ??
    - BuildClusters: **ECal (energy to be filled, positions to be filled), SAC (energy to be filled, positions to be filled), Target (calibration and positions are nominal, to be further calibrated), PVeto, EVeto, HEPVeto**
    - BuildHits: available for all detectors
      - basic (known) calibration constants, basic (known) cabling fixes … when can they be put in production ?
  - DQ flagging of the detector: how
  - ….

BACKUP



```cpp
class PadmeVReconstruction : public PadmeVNamedModule, public RecoVChannelID
{
public:

  PadmeVReconstruction(TFile*, TString, TString);
  virtual ~PadmeVReconstruction();
  //virtual TRecoVEvent* ProcessEvent(TDetectorVEvent* = 0, Event* = 0) = 0;
  virtual void ProcessEvent(TMCVEvent* = 0,TMCEvent* = 0);
  virtual void ProcessEvent(TRawEvent* = 0);
  virtual void ProcessEvent(TRecoVObject* =0, TRecoEvent* =0);
  virtual void ClearHits();
  virtual void ClearClusters();
  virtual void BuildHits(TRawEvent*);
  virtual void ReadHits(TRecoVObject*, TRecoEvent*);
  virtual void BuildClusters();

  virtual void AnalyzeEvent(TRawEvent* = 0);
  virtual void Init(PadmeVReconstruction*);
  virtual void EndProcessing(); ///< Call from derived classes
  virtual void ParseConfFile(TString);
  virtual void HistoInit();
  virtual void HistoExit();
  virtual void AddHisto(string,TH1 *);
  virtual TH1* GetHisto(string);

  static void Exception(TString);

public:

  //TRecoVEvent* GetRecoEvent() { return fRecoEvent; };
  //void         SetRecoEvent(TRecoVEvent* value) { fRecoEvent = value; };

  PadmeVReconstruction* GetMainReco() { return fMainReco; };
  void                  SetMainReco(PadmeVReconstruction* value) { fMainReco = value; };

  TFile* GetHistoFile() { return fHistoFile; };

  TString GetConfigFileName() { return fConfigFileName; };
  void    SetConfigFileName(TString val) { fConfigFileName = val; };
  utl::ConfigParser *GetConfigParser(){return fConfigParser;};
  PadmeVRecoConfig *GetConfig(){return fConfig;};
  vector<TRecoVHit *> &GetRecoHits(){return fHits;};
  vector<TRecoVCluster *> &GetClusters(){return fClusters;}


  // Use to get an existing directory or create if not already made
  //TDirectory* GetOrMakeDir(TDirectory *inDir,TString dirName);

protected:

  TFile* fHistoFile;
  PadmeVReconstruction* fMainReco;

  //TRecoVEvent * fRecoEvent;

  TString fConfigFileName;
  utl::ConfigParser *fConfigParser;
  PadmeVRecoConfig *fConfig;


  map<string,TH1 *> fHistoMap;

  vector<TRecoVHit *> fHits;
  vector<TRecoVCluster *> fClusters;

  ChannelVReco *fChannelReco;
  PadmeVCalibration *fChannelCalibration;
};
```

✓ feature/ClusterReco2

new

reconstruction from reco-hits

new

common interface to build clusters

the base reconstruction class ***PadmeVReconstruction*** in PadmeReco/RecoBase

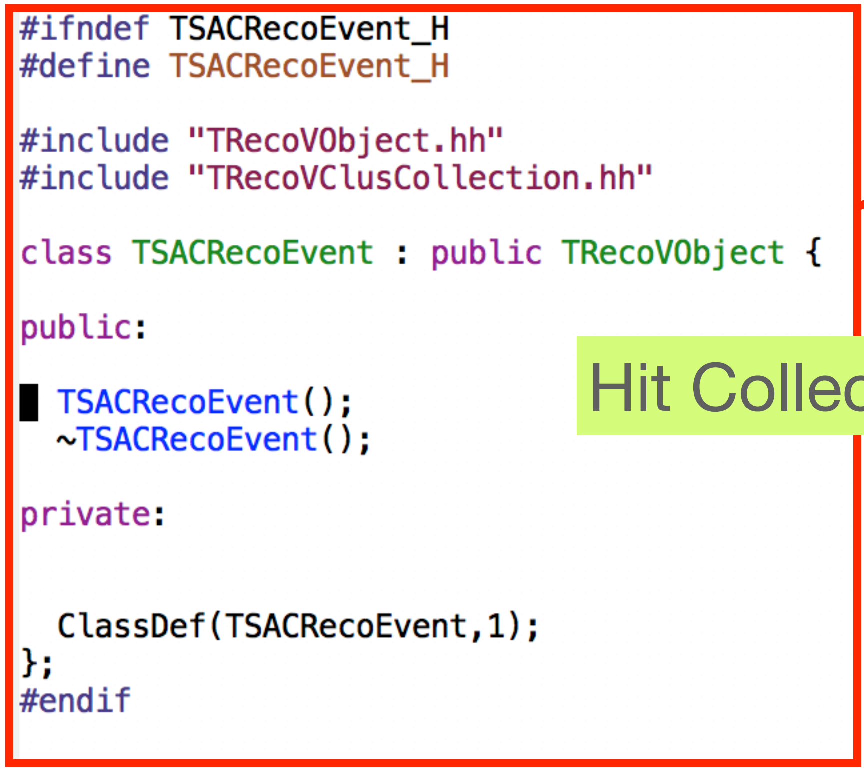
new

common interface to retrieve clusters

new

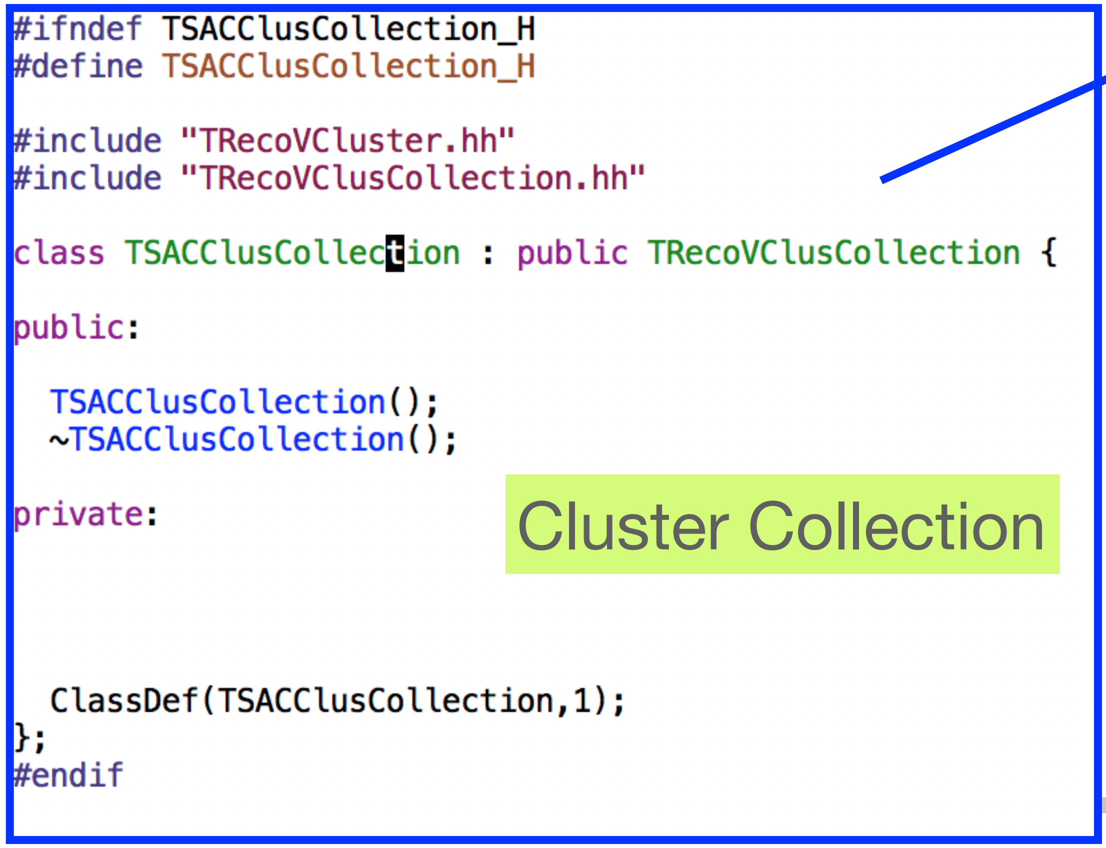
vector of pointers to clusters

CLASSES CORRESPONDING TO BRANCHES



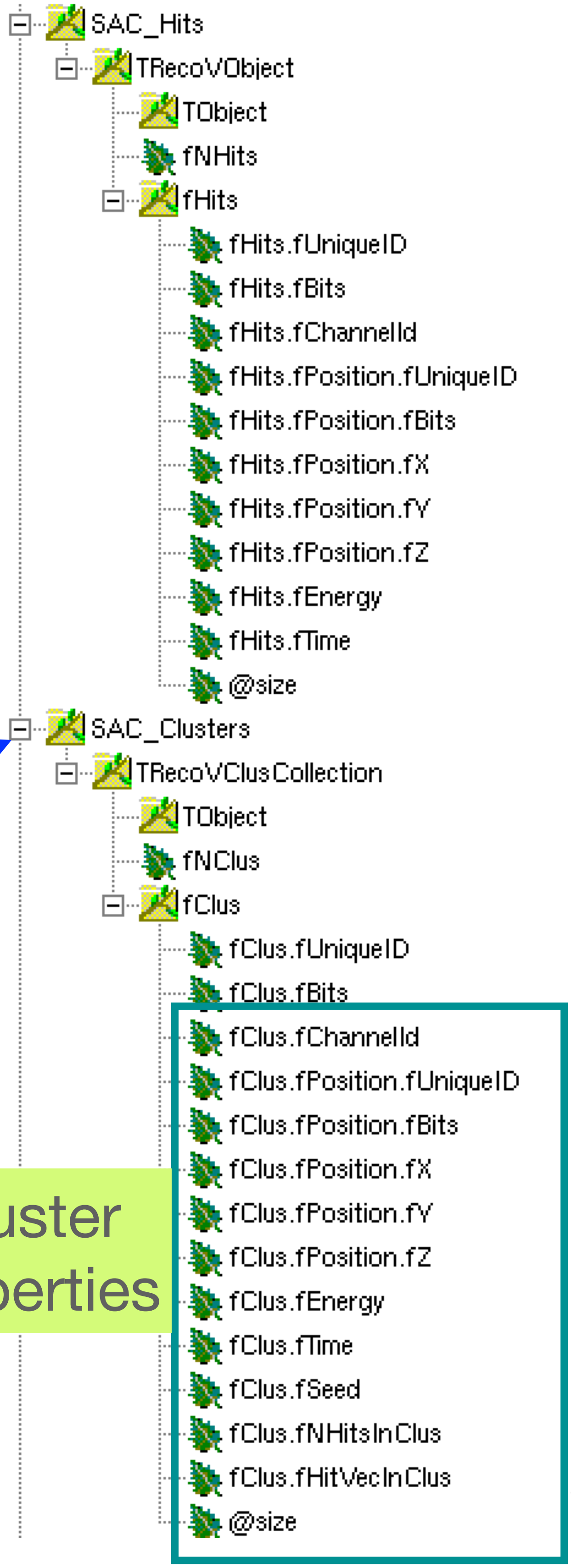
```
#ifndef TSACRecoEvent_H
#define TSACRecoEvent_H

#include "TRecoVObject.hh"
#include "TRecoVClusCollection.hh"

class TSACRecoEvent : public TRecoVObject {

public:

  TSACRecoEvent();
  ~TSACRecoEvent();

private:


  ClassDef(TSACRecoEvent,1);
};
#endif
```

Hit Collection



```
#ifndef TSACClusCollection_H
#define TSACClusCollection_H

#include "TRecoVCluster.hh"
#include "TRecoVClusCollection.hh"

class TSACClusCollection : public TRecoVClusCollection {

public:

  TSACClusCollection();
  ~TSACClusCollection();

private:



  ClassDef(TSACClusCollection,1);
};
#endif
```

Cluster Collection



```
SAC_Hits
  TRecoVObject
    TObject
    fNHits
    fHits
      fHits.fUniqueID
      fHits.fBits
      fHits.fChannelId
      fHits.fPosition.fUniqueID
      fHits.fPosition.fBits
      fHits.fPosition.fX
      fHits.fPosition.fY
      fHits.fPosition.fZ
      fHits.fEnergy
      fHits.fTime
      @size
SAC_Clusters
  TRecoVClusCollection
    TObject
    fNClus
    fClus
      fClus.fUniqueID
      fClus.fBits
      fClus.fChannelId
      fClus.fPosition.fUniqueID
      fClus.fPosition.fBits
      fClus.fPosition.fX
      fClus.fPosition.fY
      fClus.fPosition.fZ
      fClus.fEnergy
      fClus.fTime
      fClus.fSeed
      fClus.fNHitsInClus
      fClus.fHitVecInClus
      @size
```

Cluster properties

Ch Id of the seed

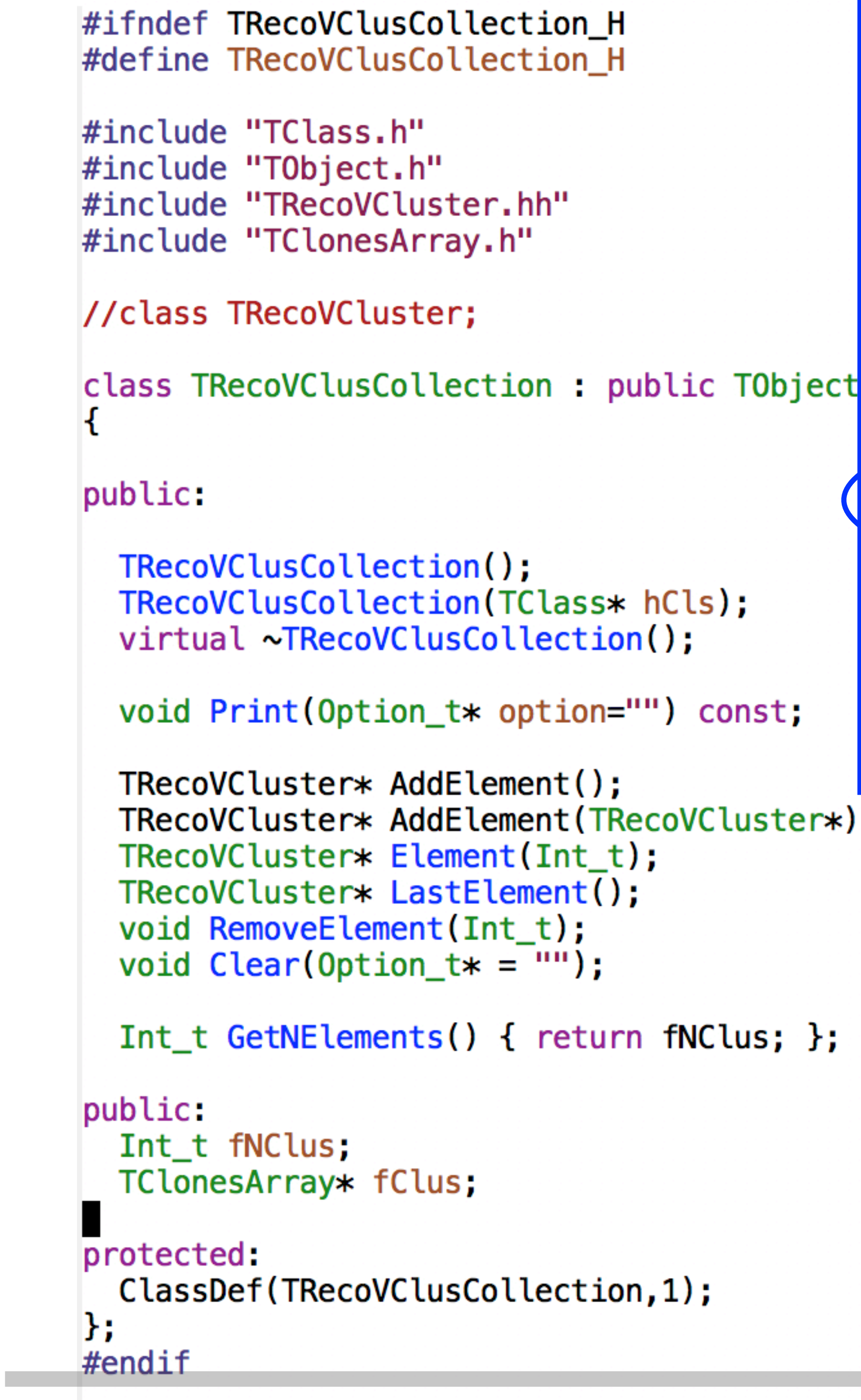
Index of the seed hit

n. hits in cluster

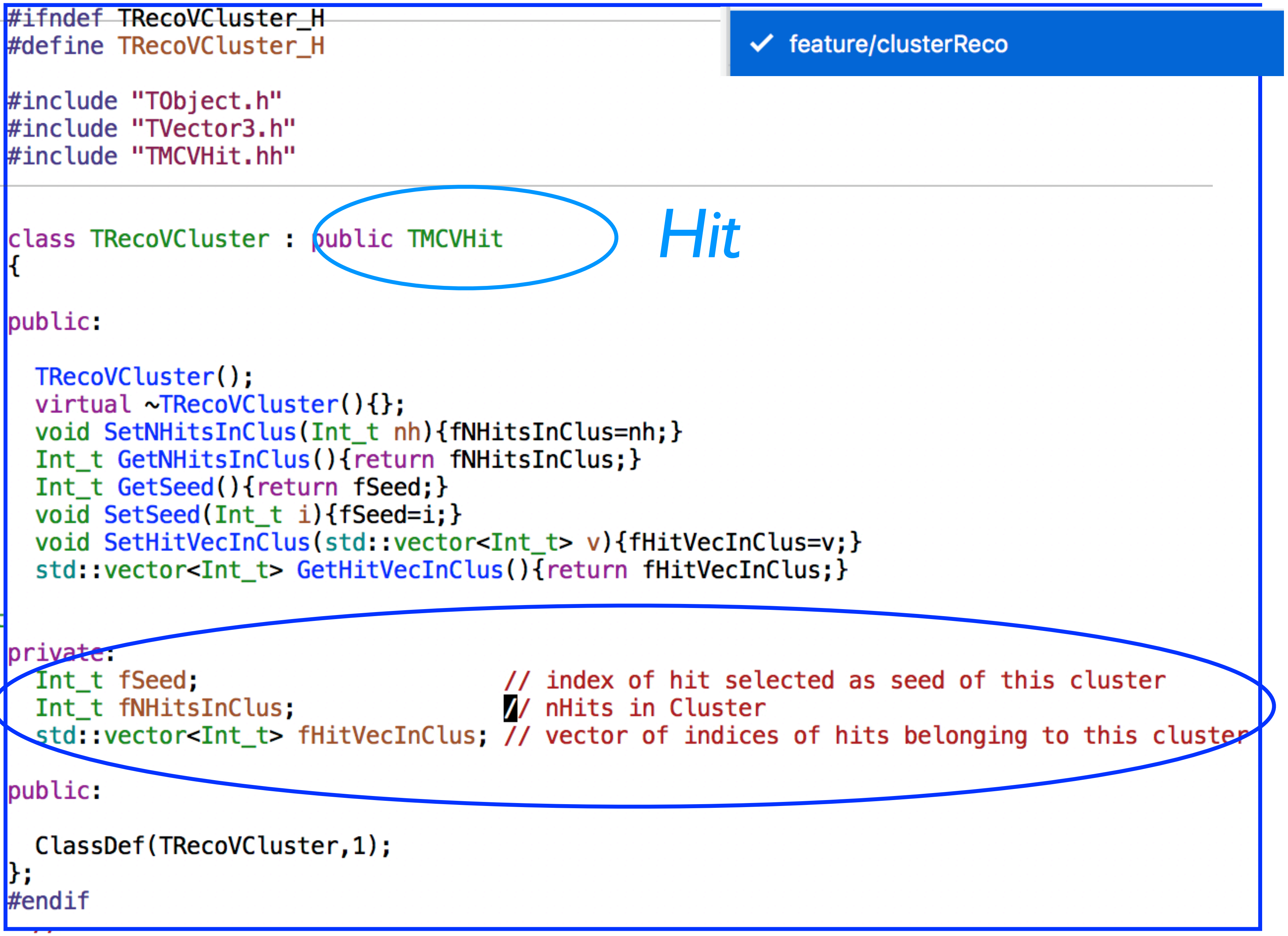
array of indices of hits in the cluster

NEW CLASSES

Collection of Clusters



```
#ifndef TRecoVClusCollection_H
#define TRecoVClusCollection_H

#include "TClass.h"
#include "TObject.h"
#include "TRecoVCluster.hh"
#include "TClonesArray.h"

//class TRecoVCluster;

class TRecoVClusCollection : public TObject
{

public:

  TRecoVClusCollection();
  TRecoVClusCollection(TClass* hCls);
  virtual ~TRecoVClusCollection();

  void Print(Option_t* option="") const;

  TRecoVCluster* AddElement();                //
  TRecoVCluster* AddElement(TRecoVCluster*); //
  TRecoVCluster* Element(Int_t);             //
  TRecoVCluster* LastElement();              //
  void RemoveElement(Int_t);                 //
  void Clear(Option_t* = "");                //

  Int_t GetNElements() { return fNClus; };

public:
  Int_t fNClus;
  TClonesArray* fClus;

protected:
  ClassDef(TRecoVClusCollection,1);
};
#endif
```



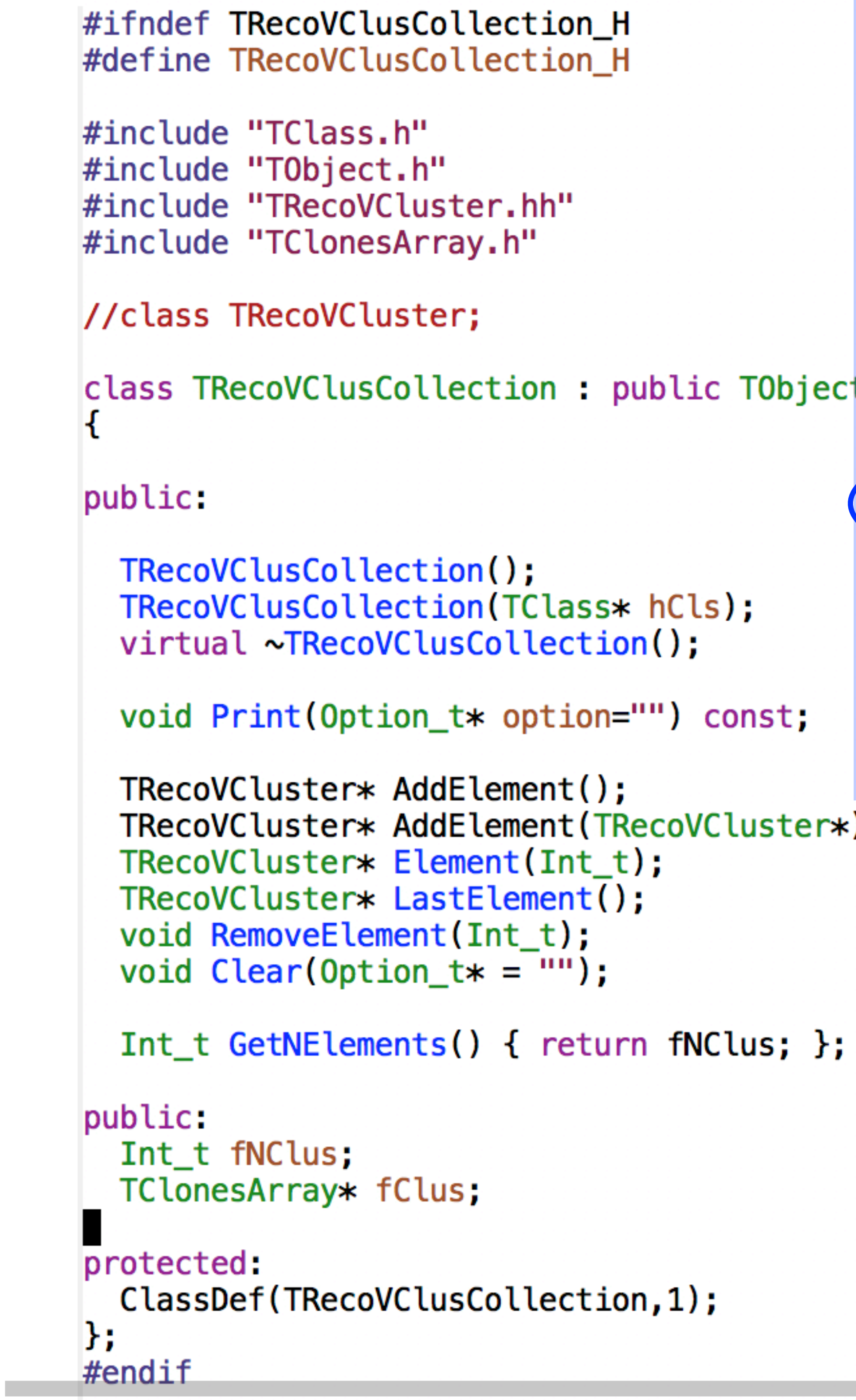
```
#ifndef TRecoVCluster_H
#define TRecoVCluster_H

#include "TObject.h"
#include "TVector3.h"
#include "TMCVHit.hh"


class TRecoVCluster : public TMCVHit
{

public:

  TRecoVCluster();
  virtual ~TRecoVCluster(){};
  void SetNHitsInClus(Int_t nh){fNHitsInClus=nh;}
  Int_t GetNHitsInClus(){return fNHitsInClus;}
  Int_t GetSeed(){return fSeed;}
  void SetSeed(Int_t i){fSeed=i;}
  void SetHitVecInClus(std::vector<Int_t> v){fHitVecInClus=v;}
  std::vector<Int_t> GetHitVecInClus(){return fHitVecInClus;}


private:
  Int_t fSeed;                          // index of hit selected as seed of this cluster
  Int_t fNHitsInClus;                   // nHits in Cluster
  std::vector<Int_t> fHitVecInClus;     // vector of indices of hits belonging to this cluster

public:

  ClassDef(TRecoVCluster,1);
};
#endif
```

Hit

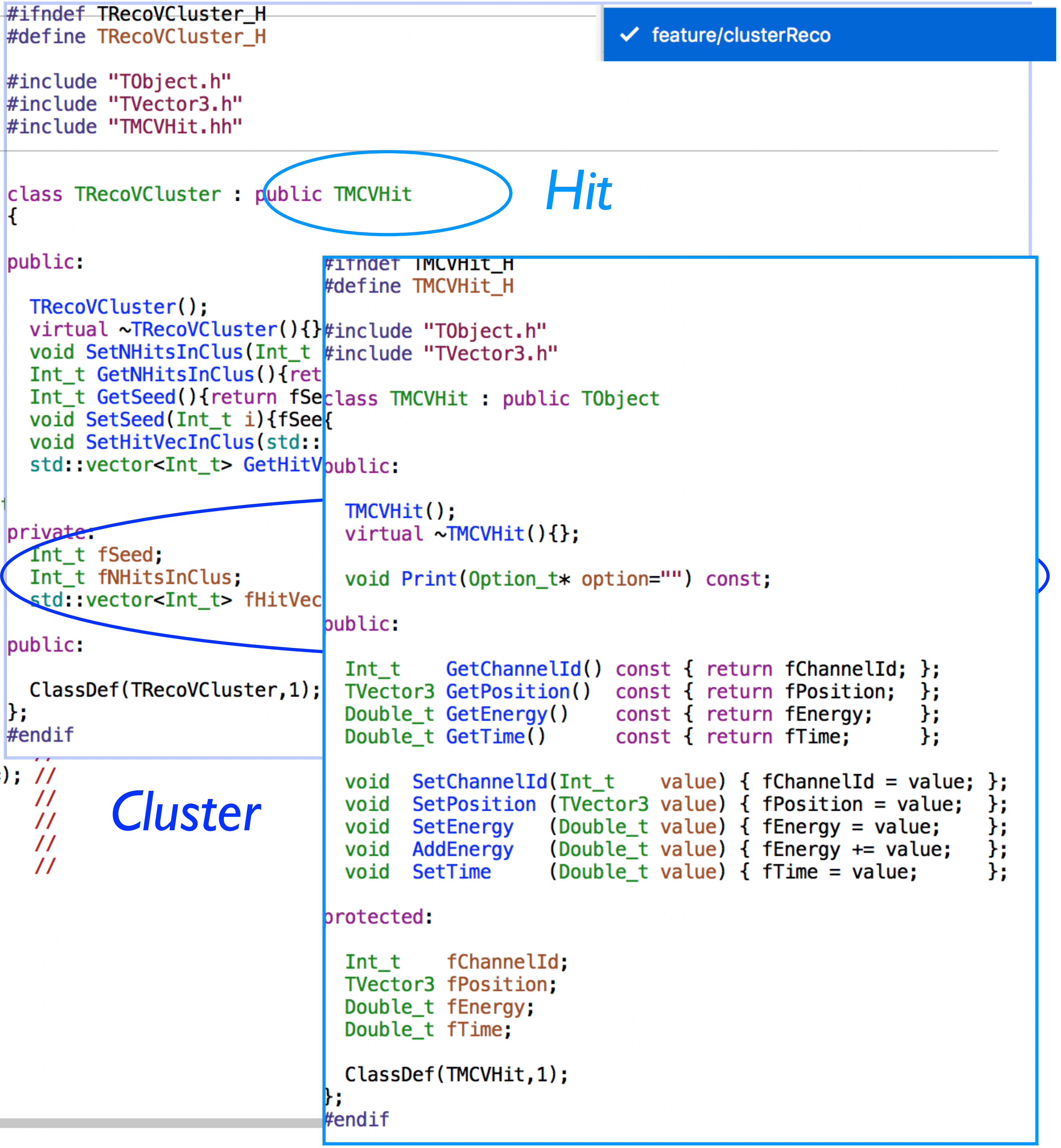
Cluster

New Classes

Collection of Clusters



```
#ifndef TRecoVClusCollection_H
#define TRecoVClusCollection_H

#include "TClass.h"
#include "TObject.h"
#include "TRecoVCluster.hh"
#include "TClonesArray.h"

//class TRecoVCluster;

class TRecoVClusCollection : public TObject
{

public:

  TRecoVClusCollection();
  TRecoVClusCollection(TClass* hCls);
  virtual ~TRecoVClusCollection();

  void Print(Option_t* option="") const;

  TRecoVCluster* AddElement();
  TRecoVCluster* AddElement(TRecoVCluster*); //
  TRecoVCluster* Element(Int_t);             //
  TRecoVCluster* LastElement();              //
  void RemoveElement(Int_t);                 //
  void Clear(Option_t* = "");                //

  Int_t GetNElements() { return fNClus; };

public:
  Int_t fNClus;
  TClonesArray* fClus;

protected:
  ClassDef(TRecoVClusCollection,1);
};
#endif
```



Hit

Cluster

```
#ifndef TRecoVCluster_H
#define TRecoVCluster_H

#include "TObject.h"
#include "TVector3.h"
#include "TMCVHit.hh"


class TRecoVCluster : public TMCVHit
{

public:

  TRecoVCluster();
  virtual ~TRecoVCluster(){}
  void SetNHitsInClus(Int_t
  Int_t GetNHitsInClus(){ret
  Int_t GetSeed(){return fSe
  void SetSeed(Int_t i){fSee
  void SetHitVecInClus(std::
  std::vector<Int_t> GetHitV


private:
  Int_t fSeed;
  Int_t fNHitsInClus;
  std::vector<Int_t> fHitVec

public:

  ClassDef(TRecoVCluster,1);
};
#endif
```

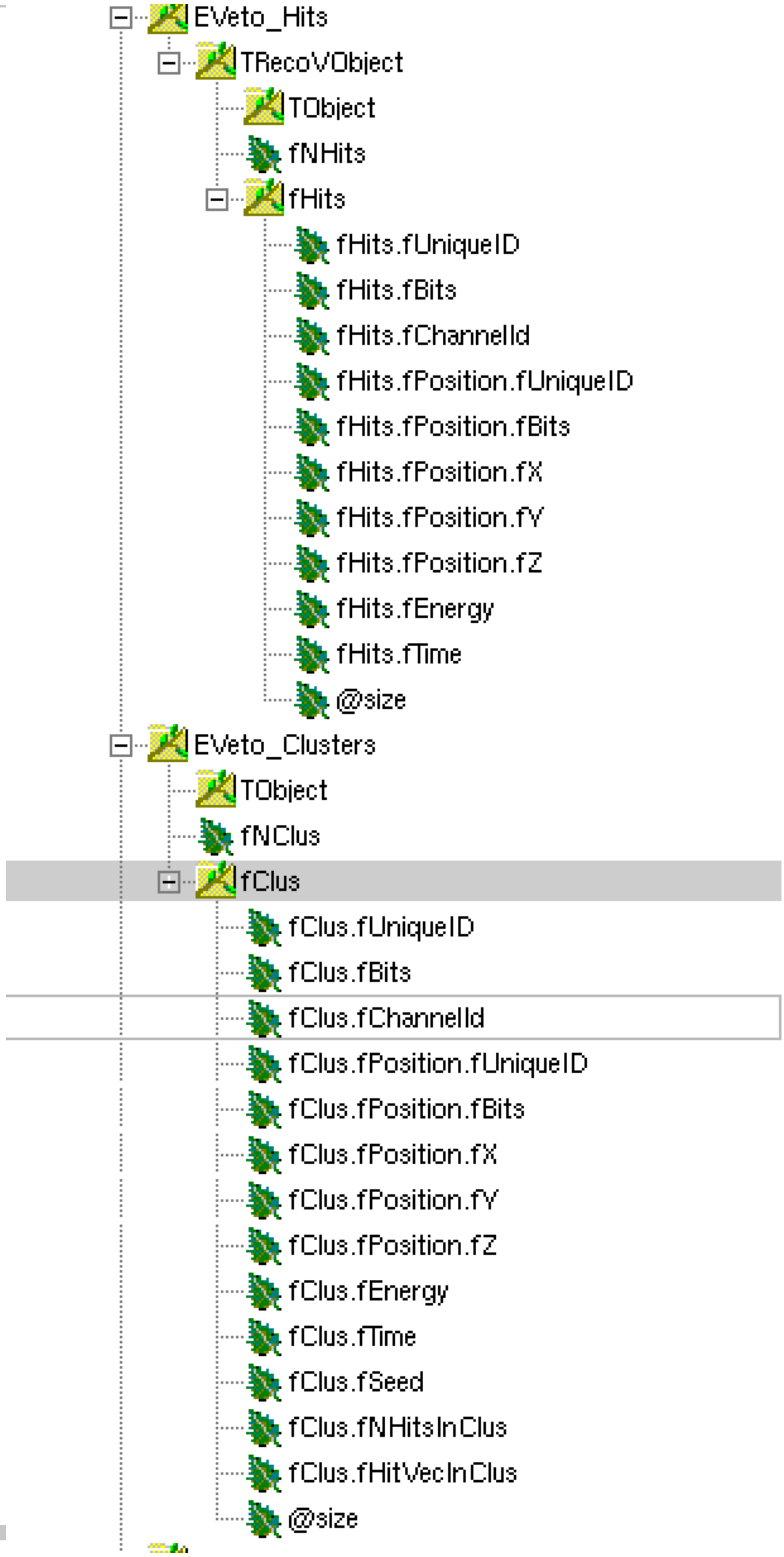
```
#ifndef TMCVHit_H
#define TMCVHit_H

#include "TObject.h"
#include "TVector3.h"

class TMCVHit : public TObject
{

public:

  TMCVHit();
  virtual ~TMCVHit(){};

  void Print(Option_t* option="") const;

public:

  Int_t     GetChannelId() const { return fChannelId; };
  TVector3 GetPosition()  const { return fPosition;  };
  Double_t GetEnergy()    const { return fEnergy;    };
  Double_t GetTime()      const { return fTime;      };

  void  SetChannelId(Int_t    value) { fChannelId = value; };
  void  SetPosition (TVector3 value) { fPosition = value;  };
  void  SetEnergy   (Double_t value) { fEnergy = value;    };
  void  AddEnergy   (Double_t value) { fEnergy += value;   };
  void  SetTime     (Double_t value) { fTime = value;      };

protected:

  Int_t     fChannelId;
  TVector3 fPosition;
  Double_t fEnergy;
  Double_t fTime;

  ClassDef(TMCVHit,1);
};
#endif
```

✓ feature/ClusterReco2

ROOT FILE (OUTPUT OF THE RECO)

- Ready for all detectors
  - In PadmeReco/RecoBase/ PadmeVReconstruction
    - BuildClusters(){;}
  - PadmeReco/src/RecoVRootIO

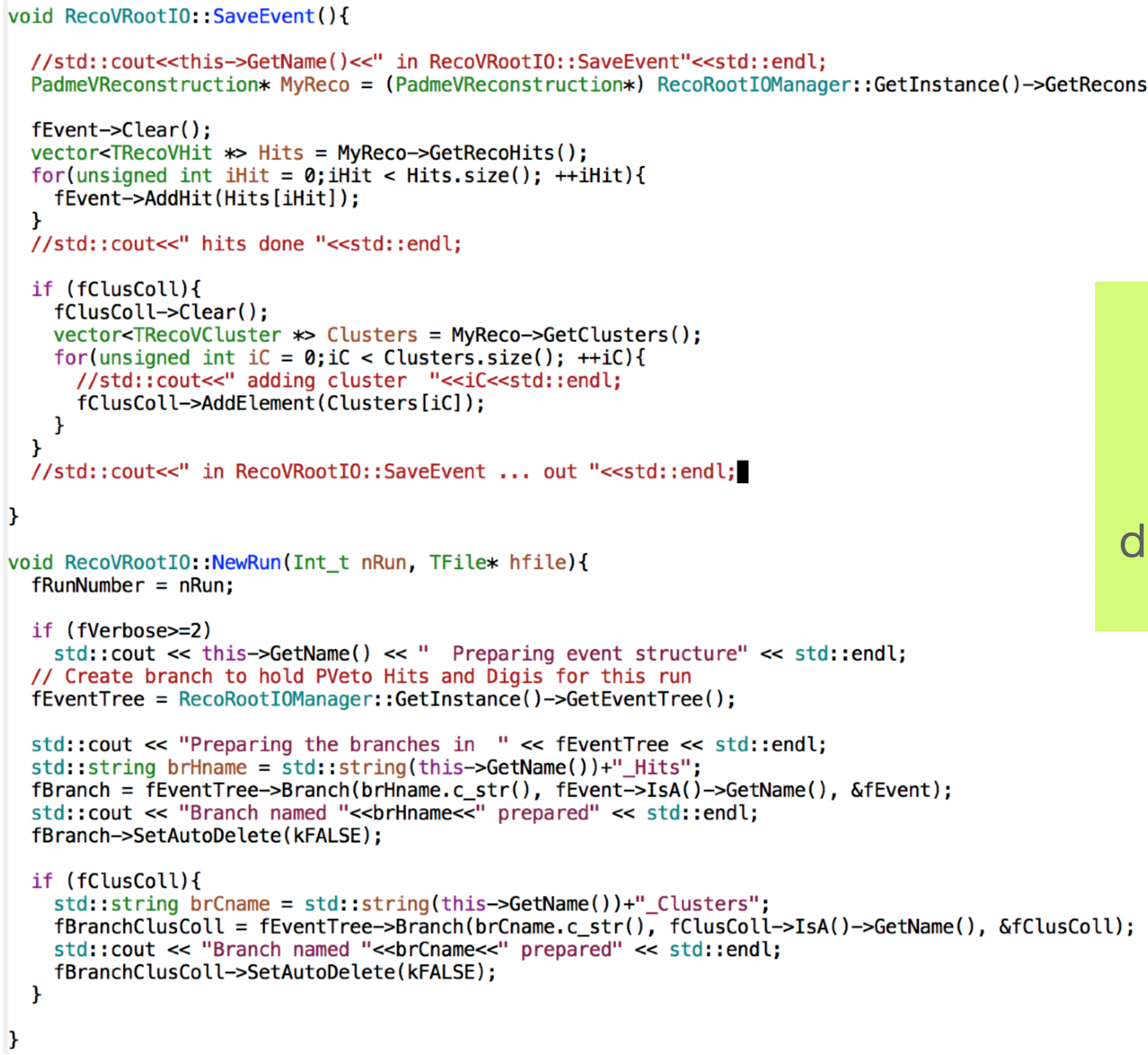
  - Clusters filled for SAC and ECAL with algorithms implemented in XXXXReconstruction::Analyze
    - XXXReconstruction::BuildClusters()



```
EVeto_Hits
    TRecoVObject
        TObject
        fNHits
        fHits
            fHits.fUniqueID
            fHits.fBits
            fHits.fChannelId
            fHits.fPosition.fUniqueID
            fHits.fPosition.fBits
            fHits.fPosition.fX
            fHits.fPosition.fY
            fHits.fPosition.fZ
            fHits.fEnergy
            fHits.fTime
            @size
EVeto_Clusters
    TObject
    fNClus
    fClus
        fClus.fUniqueID
        fClus.fBits
        fClus.fChannelId
        fClus.fPosition.fUniqueID
        fClus.fPosition.fBits
        fClus.fPosition.fX
        fClus.fPosition.fY
        fClus.fPosition.fZ
        fClus.fEnergy
        fClus.fTime
        fClus.fSeed
        fClus.fNHitsInClus
        fClus.fHitVecInClus
        @size
```

✓ feature/ClusterReco2

PERSISTENCY: WRITING TO ROOT



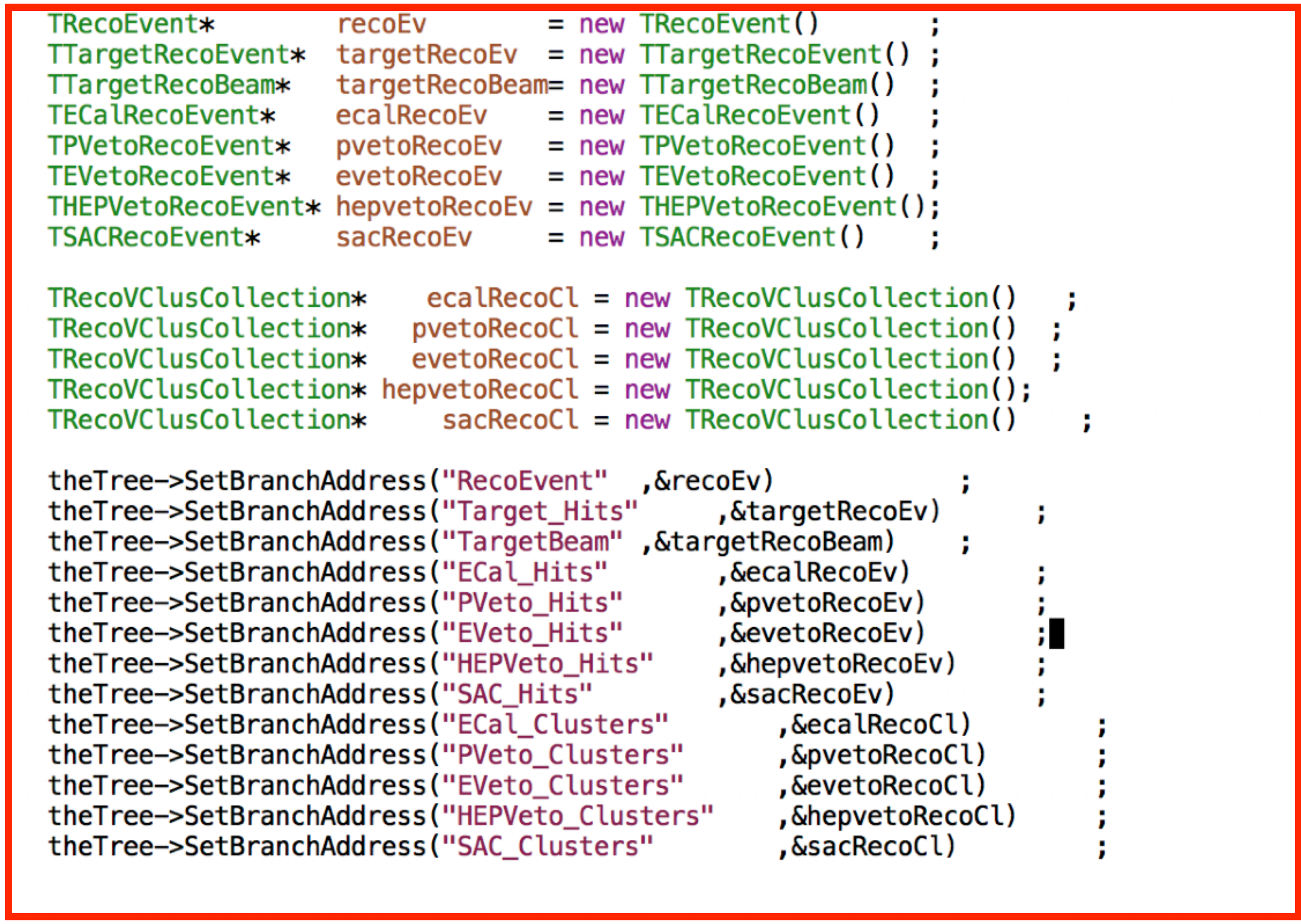
```
void RecoVRootIO::SaveEvent(){

  //std::cout<<this->GetName()<<" in RecoVRootIO::SaveEvent"<<std::endl;
  PadmeVReconstruction* MyReco = (PadmeVReconstruction*) RecoRootIOManager::GetInstance()->GetReconstruction()->FindReco(this->GetName());

  fEvent->Clear();
  vector<TRecoVHit *> Hits = MyReco->GetRecoHits();
  for(unsigned int iHit = 0;iHit < Hits.size(); ++iHit){
    fEvent->AddHit(Hits[iHit]);
  }
  //std::cout<<" hits done "<<std::endl;

  if (fClusColl){
    fClusColl->Clear();
    vector<TRecoVCluster *> Clusters = MyReco->GetClusters();
    for(unsigned int iC = 0;iC < Clusters.size(); ++iC){
      //std::cout<<" adding cluster  "<<iC<<std::endl;
      fClusColl->AddElement(Clusters[iC]);
    }
  }
  //std::cout<<" in RecoVRootIO::SaveEvent ... out "<<std::endl;

}

void RecoVRootIO::NewRun(Int_t nRun, TFile* hfile){
  fRunNumber = nRun;

  if (fVerbose>=2)
    std::cout << this->GetName() << "  Preparing event structure" << std::endl;
  // Create branch to hold PVeto Hits and Digis for this run
  fEventTree = RecoRootIOManager::GetInstance()->GetEventTree();

  std::cout << "Preparing the branches in  " << fEventTree << std::endl;
  std::string brHname = std::string(this->GetName())+"_Hits";
  fBranch = fEventTree->Branch(brHname.c_str(), fEvent->IsA()->GetName(), &fEvent);
  std::cout << "Branch named "<<brHname<<" prepared" << std::endl;
  fBranch->SetAutoDelete(kFALSE);

  if (fClusColl){
    std::string brCname = std::string(this->GetName())+"_Clusters";
    fBranchClusColl = fEventTree->Branch(brCname.c_str(), fClusColl->IsA()->GetName(), &fClusColl);
    std::cout << "Branch named "<<brCname<<" prepared" << std::endl;
    fBranchClusColl->SetAutoDelete(kFALSE);
  }

}
```

Generic:
ready for all detectors

basically when writing detector specific code you can ignore these details

Reading Reconstruction Output

- The analysis main
  - must read *many* branches
  - However access to objects is very simple (uniform for all hits, clusters)



```
TRecoEvent*        recoEv        = new TRecoEvent()       ;
TTargetRecoEvent*  targetRecoEv  = new TTargetRecoEvent() ;
TTargetRecoBeam*   targetRecoBeam= new TTargetRecoBeam()  ;
TECalRecoEvent*    ecalRecoEv    = new TECalRecoEvent()   ;
TPVetoRecoEvent*   pvetoRecoEv   = new TPVetoRecoEvent()  ;
TEVetoRecoEvent*   evetoRecoEv   = new TEVetoRecoEvent()  ;
THEPVetoRecoEvent* hepvetoRecoEv = new THEPVetoRecoEvent();
TSACRecoEvent*     sacRecoEv     = new TSACRecoEvent()    ;

TRecoVClusCollection*    ecalRecoCl = new TRecoVClusCollection()   ;
TRecoVClusCollection*   pvetoRecoCl = new TRecoVClusCollection()  ;
TRecoVClusCollection*   evetoRecoCl = new TRecoVClusCollection()  ;
TRecoVClusCollection* hepvetoRecoCl = new TRecoVClusCollection();
TRecoVClusCollection*     sacRecoCl = new TRecoVClusCollection()    ;

theTree->SetBranchAddress("RecoEvent"  ,&recoEv)              ;
theTree->SetBranchAddress("Target_Hits"     ,&targetRecoEv)       ;
theTree->SetBranchAddress("TargetBeam" ,&targetRecoBeam)      ;
theTree->SetBranchAddress("ECal_Hits"       ,&ecalRecoEv)         ;
theTree->SetBranchAddress("PVeto_Hits"      ,&pvetoRecoEv)        ;
theTree->SetBranchAddress("EVeto_Hits"      ,&evetoRecoEv)        ;
theTree->SetBranchAddress("HEPVeto_Hits"    ,&hepvetoRecoEv)      ;
theTree->SetBranchAddress("SAC_Hits"        ,&sacRecoEv)          ;
theTree->SetBranchAddress("ECal_Clusters"       ,&ecalRecoCl)         ;
theTree->SetBranchAddress("PVeto_Clusters"      ,&pvetoRecoCl)        ;
theTree->SetBranchAddress("EVeto_Clusters"      ,&evetoRecoCl)        ;
theTree->SetBranchAddress("HEPVeto_Clusters"    ,&hepvetoRecoCl)      ;
theTree->SetBranchAddress("SAC_Clusters"        ,&sacRecoCl)          ;
```