Applications with LCIO

G.Mavromanolakis, University of Cambridge

Outline

- **General**
- **Testbeam data model**
- **Data flowchart**
- **Clustering with gNIKI**

General

► . **CALICE testbeam program**

: ECAL and HCAL prototypes are under construction, testbeam program is about to start soon

: use LCIO as a persistency framework for data storage

► . **Reconstruction software**

: gNIKI: general Nodes Interlaced Klustering Implementation is an algorithm for calorimeter clustering

: user can easily link it with LCIO for IO

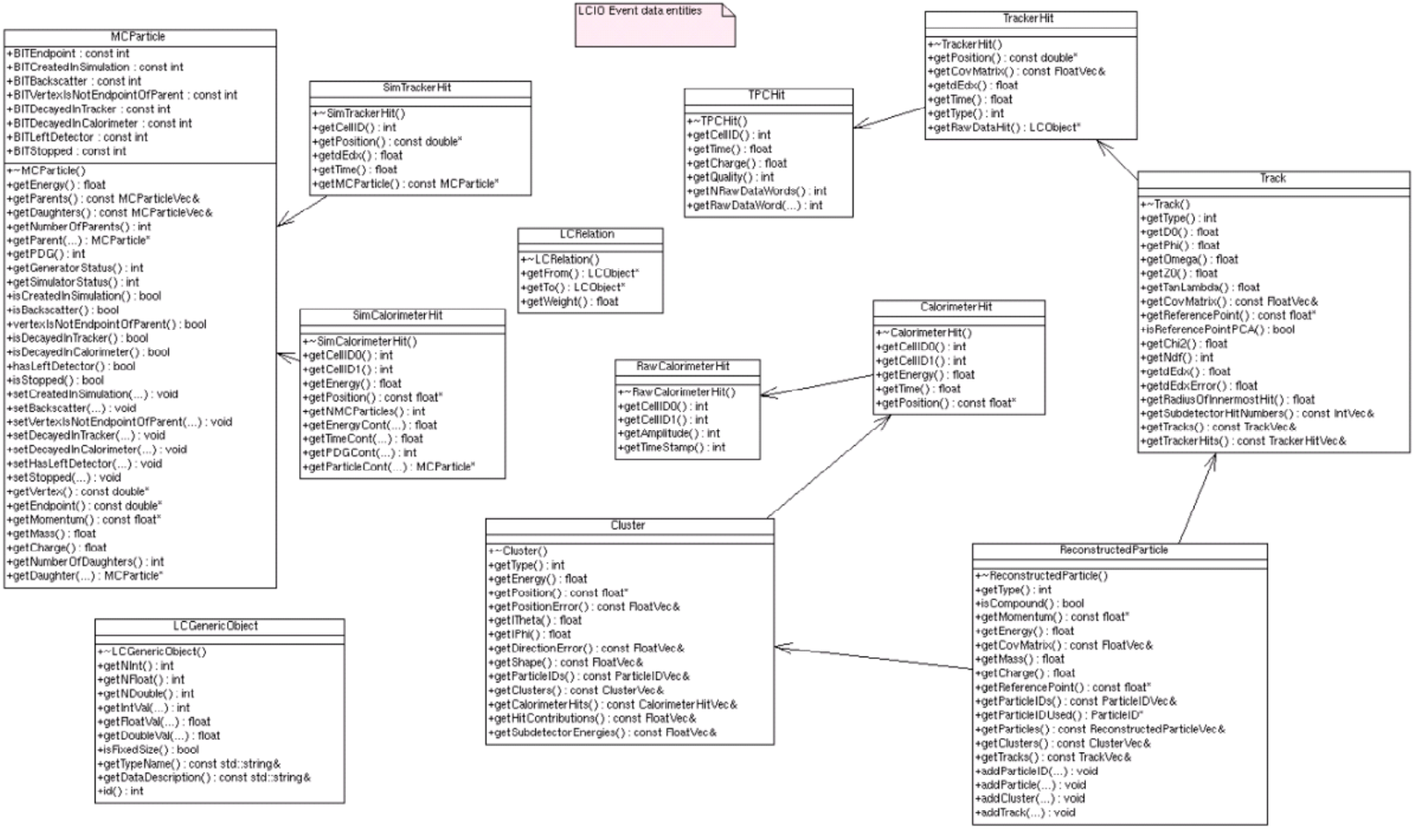
Testbeam data model

- .
  proposing a **data model** for the CALICE testbeam program
  - ▷ persistency
  - ▷ flexible implementation
  - ▷ simple user interface
  - ▷ efficiency

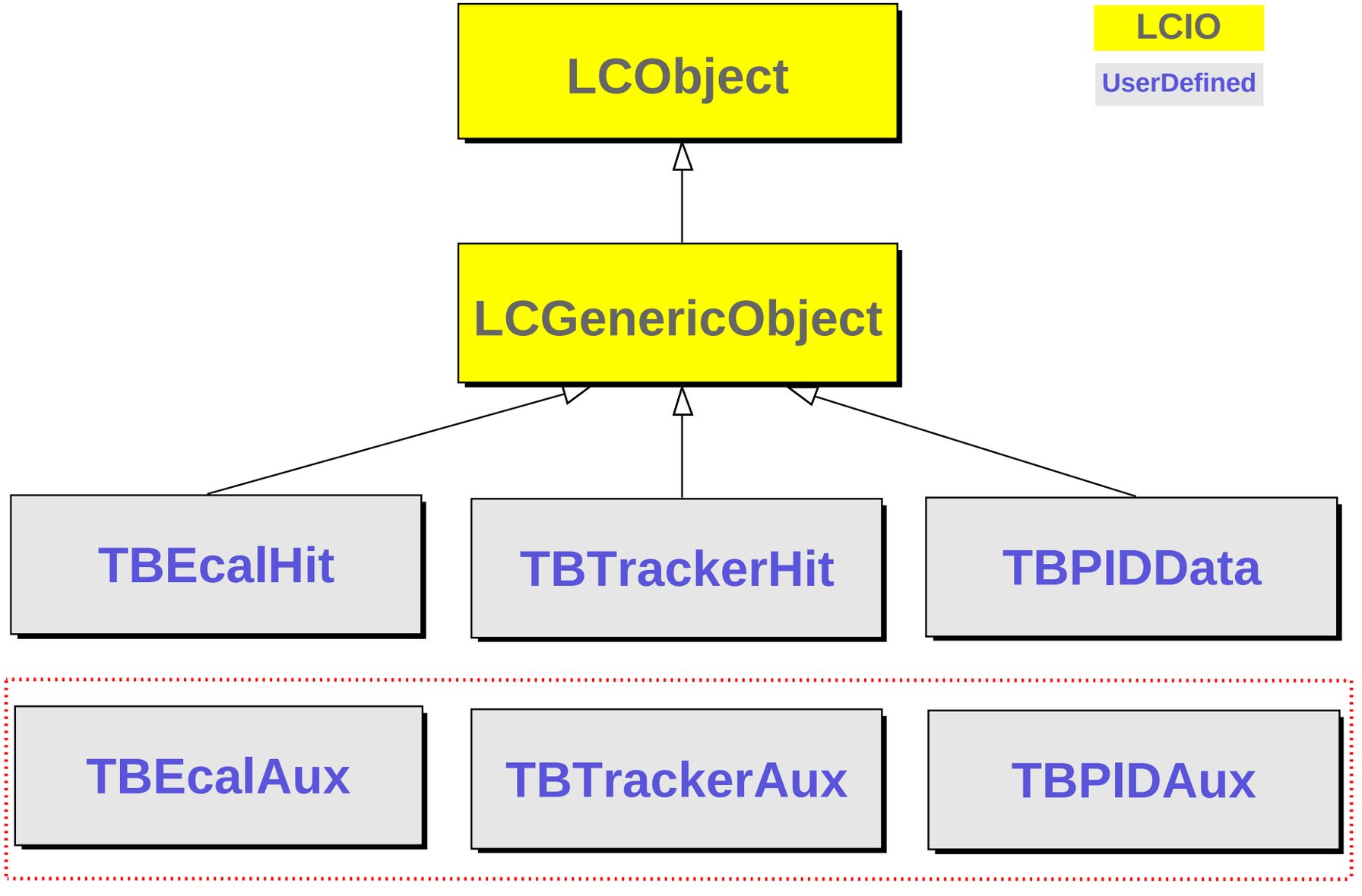
- .
  use LCIO and ROOT frameworks for some simple test implementation and benchmarking

- .
  general conversion scheme discussed
  (from raw/simulation data to analysis data)

LCIO event data entities (v01-03)



CALICE testbeam data model



example class TBEcalAux

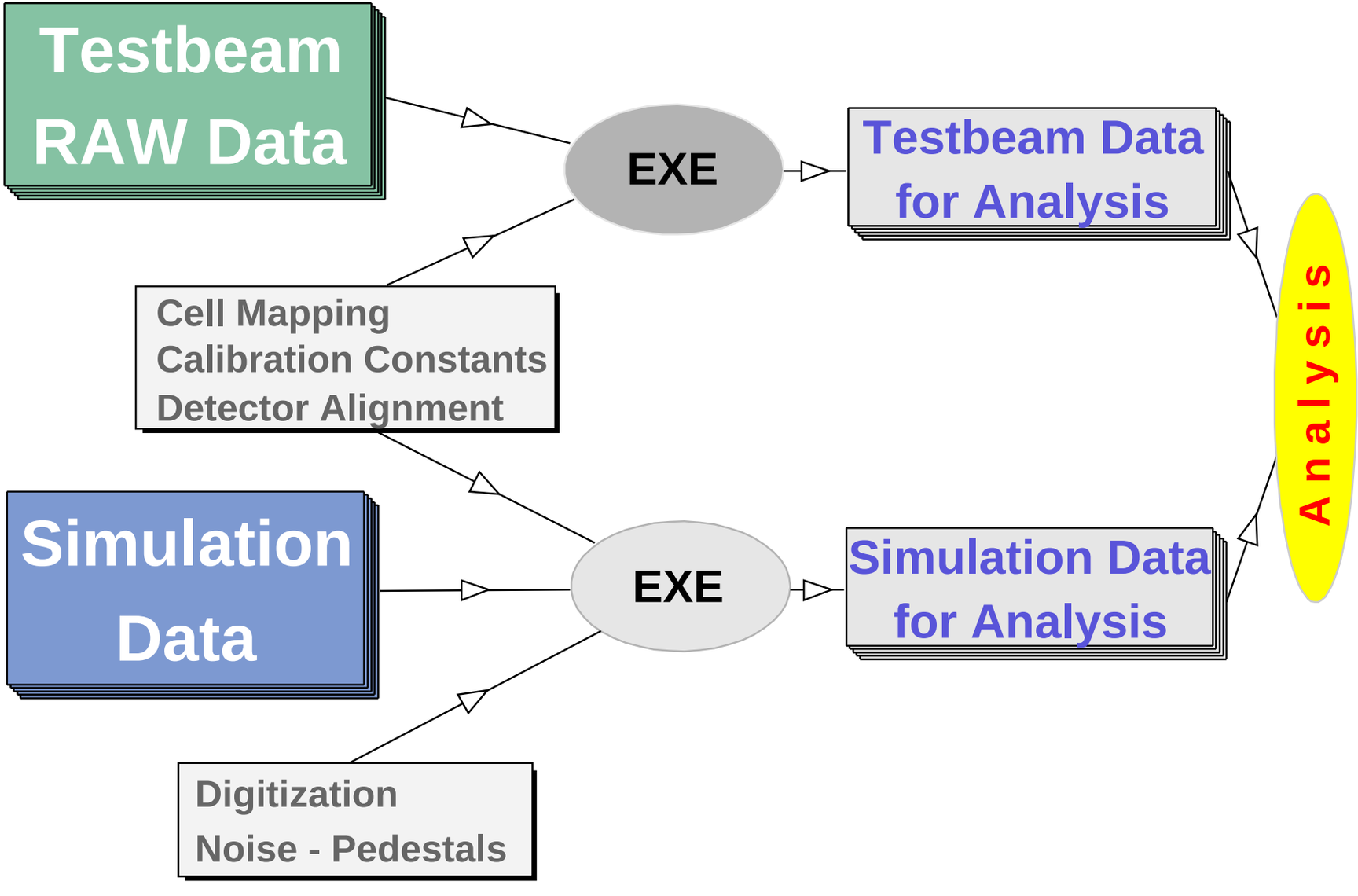
```
//////////////////////////////////////////////////////////////////////////////
//
// author     : G.Mavromanolakis
//
// description: user defined concrete class derived from LCGenericObject class
//
// comments   : uses LCIO v01-03
//
//////////////////////////////////////////////////////////////////////////////


#ifndef class_TBEcalAux
#define class_TBEcalAux

#include "lcio.h"
#include "EVENT/LCGenericObject.h"
#include "EVENT/LCObject.h"

using namespace lcio;


//////////////////////////////////////////////////////////////////////////////
class TBEcalAux : public LCGenericObject
{
       private:
                  int theK;
                  int theL;
                  int theM;
                float theX;
                float theY;
                float theZ;

       public:
                  TBEcalAux();
                  TBEcalAux(int*,float*);
                  TBEcalAux(LCObject*);
                 ~TBEcalAux();

   const static std::string TypeName;
   const static std::string DataDescription;

     const static int NumOfIntegers;
     const static int NumOfFloats;
     const static int NumOfDoubles;
           static int counter;

                  int GetK();//..... User Interface
                  int GetL();//  .
                  int GetM();//  .
                float GetX();//  .
                float GetY();//  .
                float GetZ();//...

                  int getNInt() const;             //..... LCIO Interface
                  int getNFloat() const;           //  .   (LCGenericObject
                  int getNDouble() const;          //  .    virtual methods)
                  int getIntVal(int index) const;  //  .
                float getFloatVal(int index) const; //  .
               double getDoubleVal(int index) const;//  .
                 bool isFixedSize() const;         //  .
   const std::string& getTypeName() const;         //  .
   const std::string& getDataDescription()const ;  //...

};
//////////////////////////////////////////////////////////////////////////////


#endif
```

Benchmarks

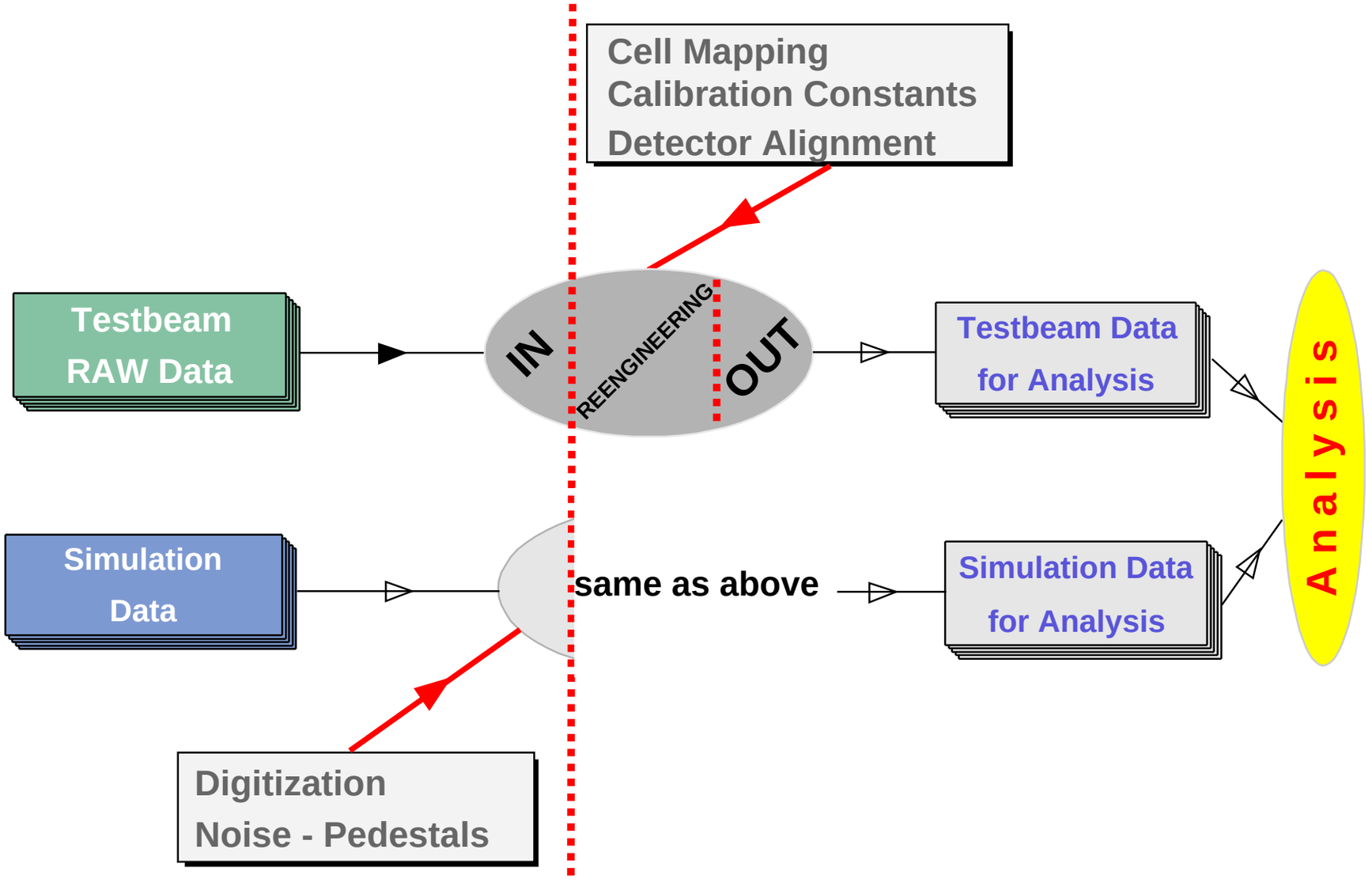
- **. configuration**
  - machine: Linux P4 2.66 GHz / 512 MB RAM
  - libs: ROOT v4.00/08 and LCIO v01-03
  - task: write/read 1 ROOT tree or 1 LCIO collection of N events $\times$ 100 hits (1 hit = 3 integers + 3 floats)

		LCIO	ROOT
100k events	**size (MB)**	**28**	**4**
	time write (sec)	**64**	**9**
	time read (sec)	**71**	**19**
500k events	**size (MB)**	**139**	**19**
	time write (sec)	**365**	**48**
	time read (sec)	**328**	**95**

Data flowchart



Data flowchart - details



Clustering with gNIKI

general Nodes Interlaced Klustering Implementation

► . **gNIKI**

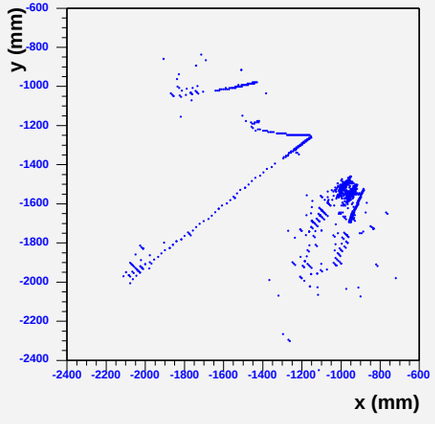
: algorithm based on minimal spanning tree theory to implement a "top-down and then bottom-up" approach to calorimeter clustering

► . **in brief**

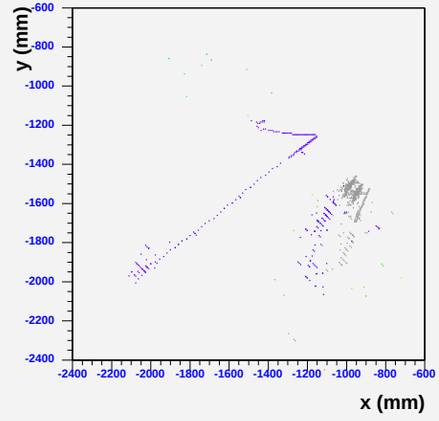
: use MST clustering algorithm with loose cut to perform coarse clustering

: then go through MST clusters found in previous step and refine using a cone-like energy flow clustering algorithm

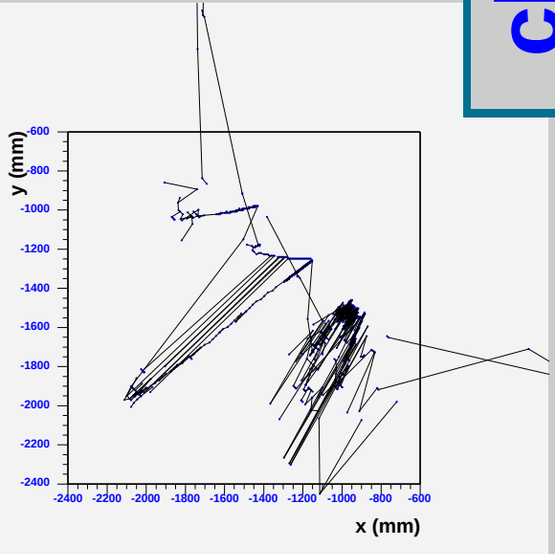
hits



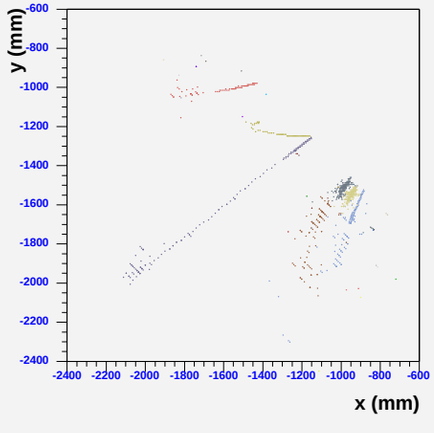
cluster MST



MST



recluster



d041130 **LC Note: ... ...**

Calorimeter clustering with gNIKI

general Nodes Interlaced Klustering Implementation

G. Mavromanolakis *

University of Cambridge, Department of Physics
Cavendish Laboratory, High Energy Physics Group
Madingley Road, Cambridge, CB3 0HE, UK

Contents

* email: gmavroma@hep.phy.cam.ac.uk or gmavroma@mail.cern.ch

source, examples, documents at:
www.hep.phy.cam.ac.uk/~gmavroma/calice/gNIKI

API

```
//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
// code    : gNIKI (general Nodes Interlaced Klustering Implementation)
// version : 0.0
// author  : G.Mavromanolakis
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

#ifndef class_MST
#define class_MST

#ifndef class_MSTclusters
#include "MSTclusters.h"
#endif

//////////////////////////////////////////////////////////////////////////////
class MST
{
  private:
                int theN;   //
                int *a_mst; //assignment
                int *b_mst; //connectivity
             double *c_mst; //point to point distance
             double *d_mst; //distance matrix
  public:
         MST();
         MST(int);
         ~MST();

                int GetNofNodes();
          const int* Read_a_mst();
          const int* Read_b_mst();
          const double* Read_c_mst();
                double* Get_d_mst();
                double* GetDistanceMatrix();
                  void PassDistanceMatrix(double*);
                  void FillDistanceMatrix();
                  void ConstructMST();
         MSTclusters* MergeClusters(float);
                  int GetNodeConnectedTo(int);
                float GetMSTLength();
                  void Dump();
          static int counter;
};
//////////////////////////////////////////////////////////////////////////////

#endif

// D E S C R I P T I O N  --  I N V E N T O R Y
//.....................................................................
// Given a set of nodes and a distance matrix the correspondi
// spanning tree is constructed by the Prim's method. Then a
// cluster analysis can be performed to merge the clusters. T
// MSTclusters.
//
// theN        : the number of points
// a_mst       : integer array with bounds [2:N]. a_mst[i]=1
//               already assigned to the tree, or 0 otherwise
// b_mst       : integer array with bounds [2:N]. b_mst[i] co
//               index of a point to which i is joined.
// c_mst       : real array with bounds [2:N] .c_mst[i] is th
//               between points i and b_mst[i].
// d_mst       : the lower triangular distance matrix with bo
//               [1:N/N-1)/2].
// MST()       : default constructor
// MST(int)    : constructor, input is theN
// ~MST()      : destructor
// GetNofNodes(): read theN
// Read_a_mst(): read array a_mst
// Read_b_mst(): read array b_mst
// Read_c_mst(): read array c_mst
// Get_d_mst() : get array d_mst
// GetDistanceMatrix(): same as Get_d_mst()
// FillDistanceMatrix(): example how to create the distance m
// GetNodeConnectedTo(int): get the point to which a point i
//                          to, input is the i
// ConstructMST(): computes the minimal spanning tree of a di
//                 by the method of Prim
// MergeClusters(float): uses the minimal spanning tree to co
//                       linkage cluster analysis, input is the thres
//                       cut the branches
// GetMSTLength(): return the total length of the mst
// Dump()      : print out arrays a_mst, b_mst, c_mst
// counter     : static counter to check that all objects are
//.....................................................................
```

API

```
//<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
// code    : gNIKI (general Nodes Interlaced Klustering Implementation)
// version : 0.0
// author  : G.Mavromanolakis
//>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>

#ifndef class_MSTclusters
#define class_MSTclusters

//////////////////////////////////////////////////////////////////////////////
class MSTclusters
{
   friend class MST;

   private:
                  int theN;
                float theCutLevel;
                  int *g_mst;
                  int *h_mst;
                  int *k_mst;
                int* Get_g_mst();
                int* Get_h_mst();
                int* Get_k_mst();

   public:
                MSTclusters();
                MSTclusters(int,float);
                ~MSTclusters();

                  int GetNofNodes();
                float GetCutLevel();
          const int* Read_g_mst();
          const int* Read_h_mst();
          const int* Read_k_mst();
          const int* GetNodes();
          const int* GetClustersIDs();
                  int GetNode(int);
                  int GetClusterIDNode(int);
                  int GetNofClusters();
                 void Dump();
          static int counter;
};
//////////////////////////////////////////////////////////////////////////////

#endif

// D E S C R I P T I O N  --  I N V E N T O R Y
//.....................................................................
// Class to hold objects created by MST::MergeClusters. See also c
//
// theN        : the number of points
// theCutLevel : the distance cut to merge clusters
// g_mst       : integer array with the point index
// h_mst       : h_mst[i]=0 if g_mst[i] belongs to the same cluste
//               if h_mst[i]!=1 then g_mst[i] is the last member of
// k_mst       : integer array with the cluster id that point g_ms
// Get_g_mst() : get array g_mst
// Get_h_mst() : get array h_mst
// Get_k_mst() : get array k_mst
// MSTclusters(): default constructor
// MSTclusters(int,float): constructor, input theN of nodes and th
// ~MSTclusters(): destructor
// GetNofNodes() : read theN
// GetCutLevel() : read theCutLevel
// Read_g_mst(): read array g_mst
// Read_h_mst(): read array h_mst
// Read_k_mst(): read array k_mst
// GetNodes(): same as Read_g_mst()
// GetClustersIDs(): same as Read_k_mst()
// GetNode(int): input is i, returns g_mst[i]
// GetClusterIDNode(int): input is i, returns k_mst[i]
// GetNofClusters(): read number of clusters found by single linka
// Dump()      : print out arrays g_mst, h_mst, k_mst
// counter     : static counter to check that all objects are dele
//.....................................................................
```

example LCIO

```
//.....................................................................
#include "MST.h"
#include "MSTclusters.h"
#include "TCluster.h"
#include "TNode.h"
#include "TCriteria.h"
//.....................................................................

void DrawView();
void Draw(TCluster*,char*,char*);
.
.
.

//////////////////////////////////////////////////////////////////////////////
int main(int argc, char** argv)
{//////////////////////////////////////////////////////////////////////////////
.
.
.
   FileName     = argv[1];
   RunNumber    = atoi( argv[2] );
   EventNumStart = atoi( argv[3] );
   EventNumEnd   = atoi( argv[4] );

   //open lcio file
   LCReader* lcReader = LCFactory::getInstance()->createLCReader();
   LCEvent* evt(0);
   lcReader->open(FileName);

   LCRunHeader* runHdr = lcReader->readNextRunHeader();
   cout << "RunNumber = " << runHdr->getRunNumber() << endl;

   //Create some histograms
.
.
.

   //Clustering criteria
   TCriteria *cuts = new TCriteria();
   cuts->SetMSTCutLevel(2.5*0.012*100);//for MST clustering
   cuts->SetReclusteringCut(0.95);
   cuts->SetRange(150.);   //mm
   cuts->SetPitchRange(9.);//mm
   cuts->SetPitchRest(30.);//mm
   cuts->Dump();

   //Loop over events
   for(int iEvent=EventNumStart; iEvent<=EventNumEnd; iEvent++)
   {//start loop over events

      int evtNumber = iEvent;
      evt = lcReader->readEvent(RunNumber, evtNumber) ;

      //loop over collections
      typedef const std::vector <std::string> StringVec ;
      StringVec* strVec = evt->getCollectionNames() ;
      for( StringVec::const_iterator name = strVec->begin(); name != strVec->end()
; name++)
      {//start loop over collections

         LCCollection* col = evt->getCollection( *name ) ;

         int NofNodes = col->getNumberOfElements();

         MST *theMST = new MST(NofNodes);

         //Fill Distance Matrix d_mst : lower triangular matrix with
         //                             diagonal elements omitted
```

```
.
.
.
         //Construct MST
         theMST->ConstructMST();

         //Merge MST clusters
         MSTclusters *theMSTclusters;
         float flevel = cuts->GetMSTCutLevel();
         theMSTclusters = theMST->MergeClusters(flevel);

         //Loop over points of this event and fill theEventCluster
         TCluster *theEventCluster = new TCluster(NofNodes);
         theEventCluster->InitReclusterCounter();
         TNode *Node = theEventCluster->GetListOfNodes();
.
.
.

         //Loop over MSTclusters of this event
         int nMSTClusters = theMSTclusters->GetNofClusters();
         for(int i=1; i<=nMSTClusters; i++)
         {
            TCluster *currentMSTCluster = theEventCluster->GetMSTCluster(i);

            currentMSTCluster->Recluster(cuts);

            theEventCluster->UpdateWithDaughter(currentMSTCluster);

            delete currentMSTCluster;
         }//end loop over MSTclusters of this event

         //bo analysis, fill histos
.
.
.

         // Delete objects of this event
         delete theMST         ; theMST        =0;
         delete theMSTclusters ; theMSTclusters =0;
         delete theEventCluster; theEventCluster=0;

      }//end loop over collections

   }//end loop over events

   //close lcio file
   lcReader->close() ;

   //save histos
.
.
.

   return 1;
}//end main()//////////////////////////////////////////////////////////////////
```

Summary

► . **CALICE testbeam**

: use LCIO data entities asis or implement new through inheritance
and develop a flexible data model suitable for the testbeam program

: conversion scheme (from raw/sim data to analysis data)
with front/back endpoints in LCIO

► . **clustering algorithm gNIKI**

: standalone C++ code to perform calorimeter clustering

: interface with LCIO to read/write data,
to be developed further to fully use/exploit LCIO entities