

CALICE testbeam data model and preparations

G.Mavromanolakis, University of Cambridge



Outline

- ▶ **Data models**
- ▶ **Data flowchart**
- ▶ **GUI application**

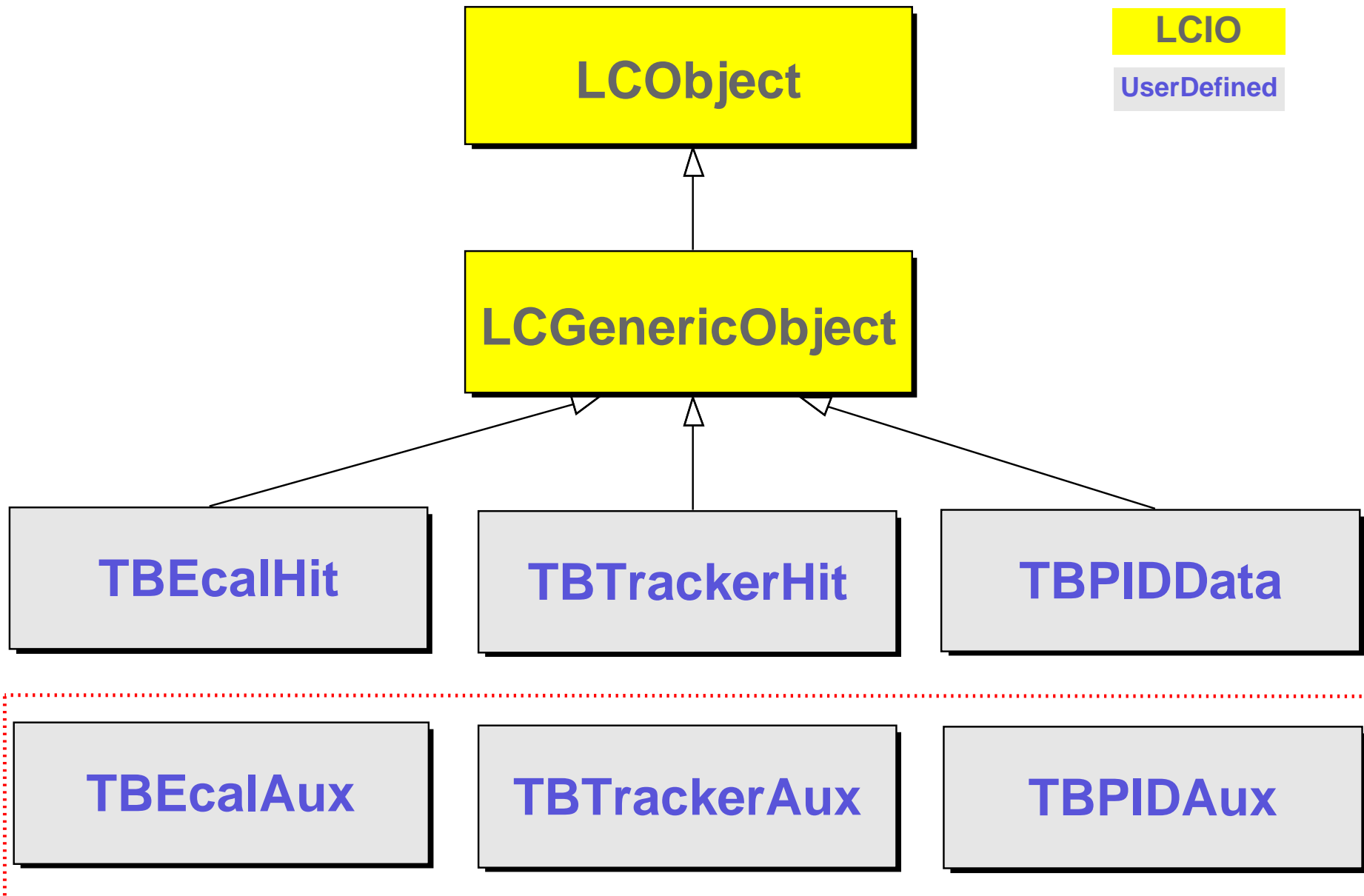
Testbeam data model

- ▶ . 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)

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;

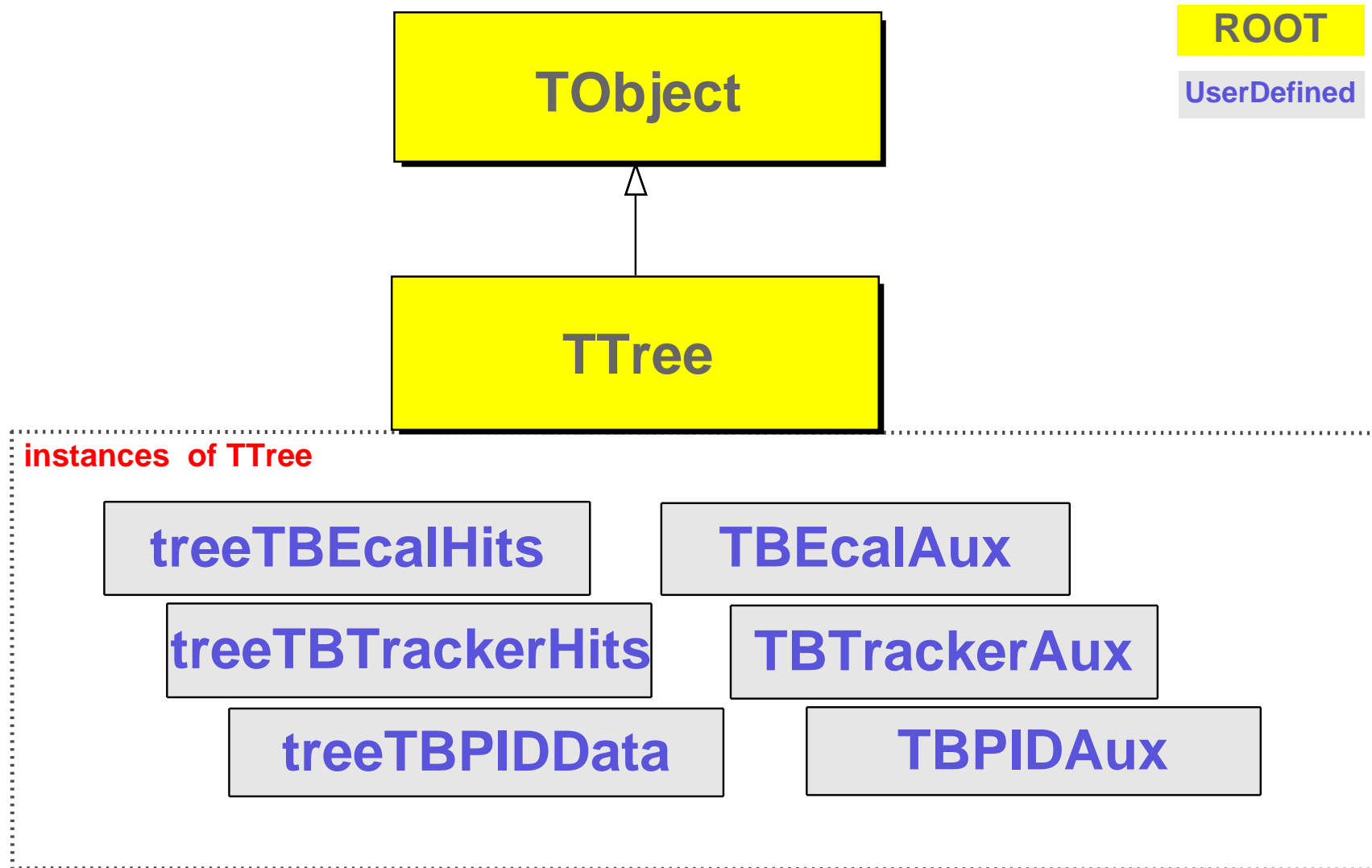
    int GetK();//.... User Interface
    int GetL();// .
    int GetM();// .
    float GetX();// .
    float GetY();// .
    float GetZ();//...

    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
```

alternative ...



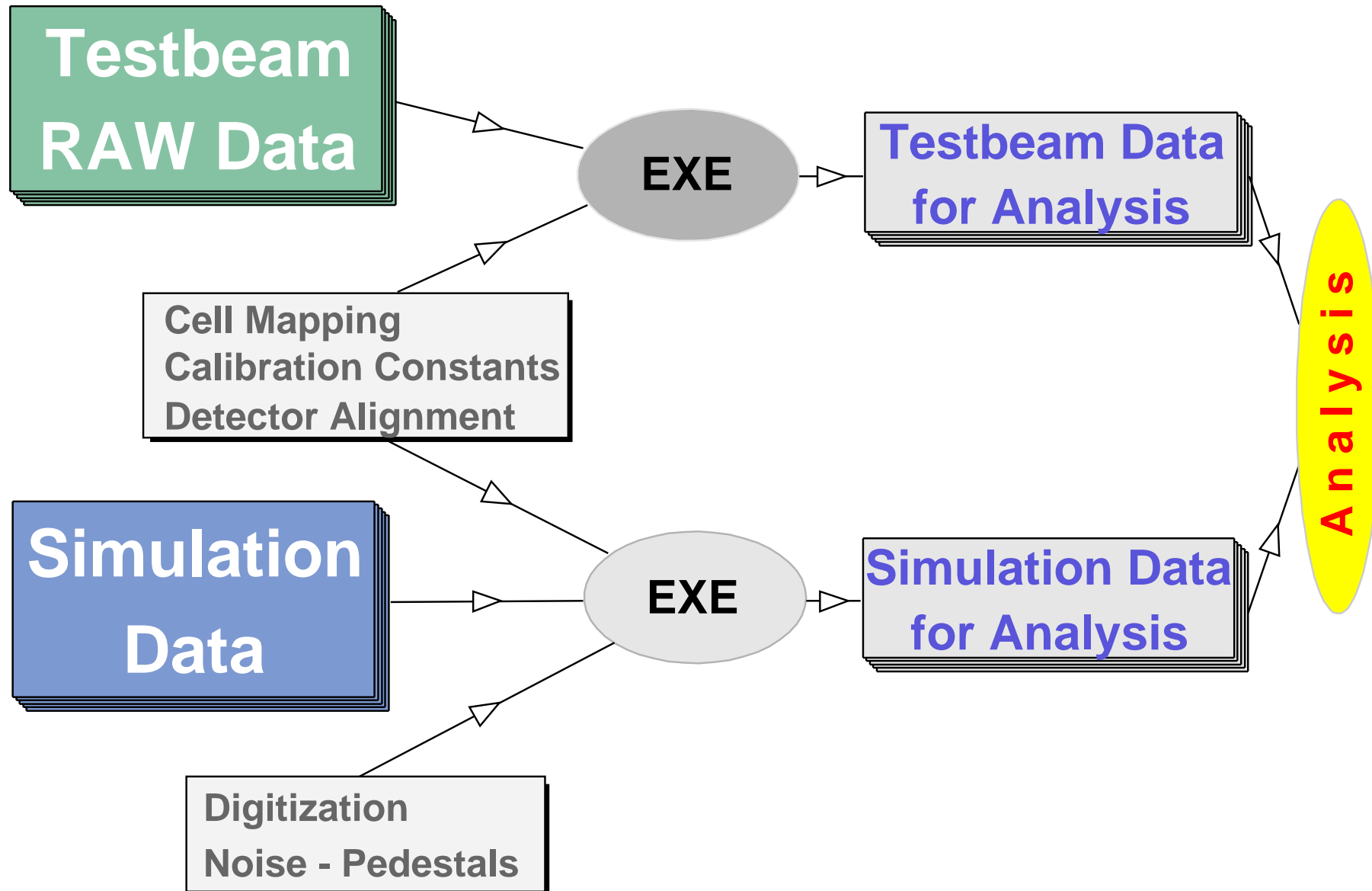
Benchmarks

► . configuration

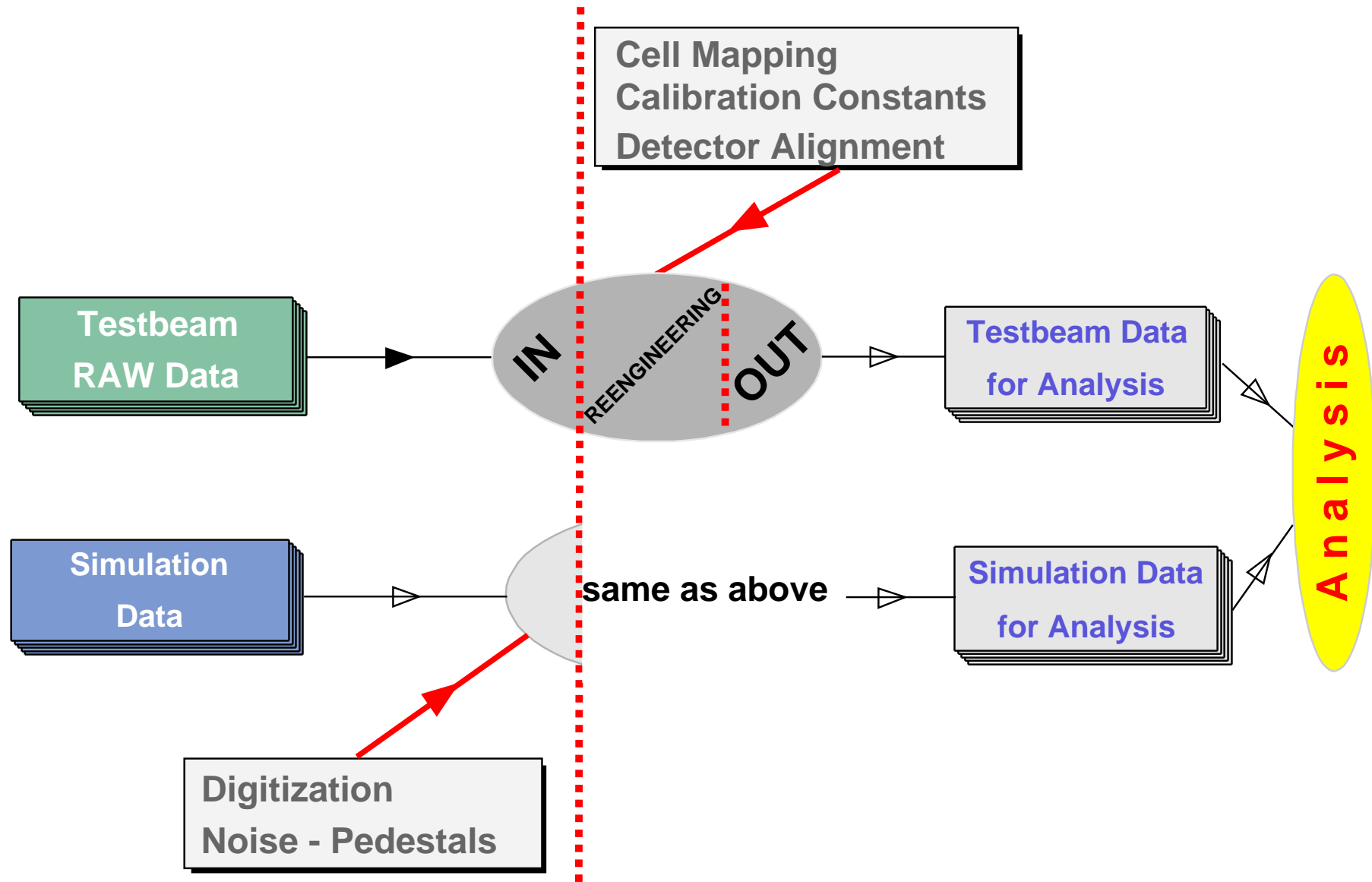
- ▷ machine: Linux P4 2.66 GHz / 512 MB RAM
- ▷ libs: ROOT v4.00/08 and LCIO v01-03
- ▷ task: write/read 1 ROOT tree or 1 LCIO collection of N events \times 100 hits (1 hit = 3 integers + 3 floats)

		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



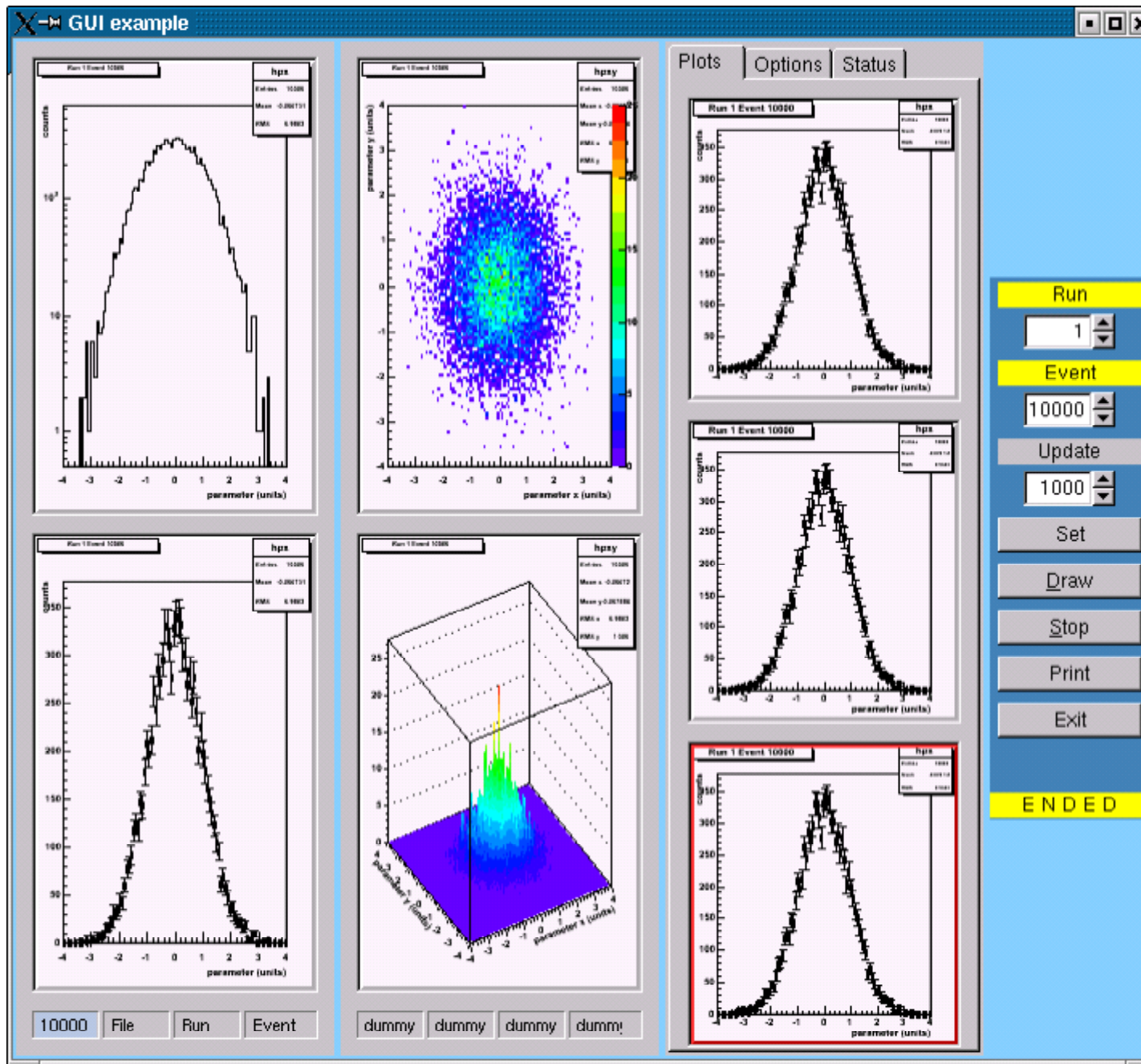
Data flowchart - details



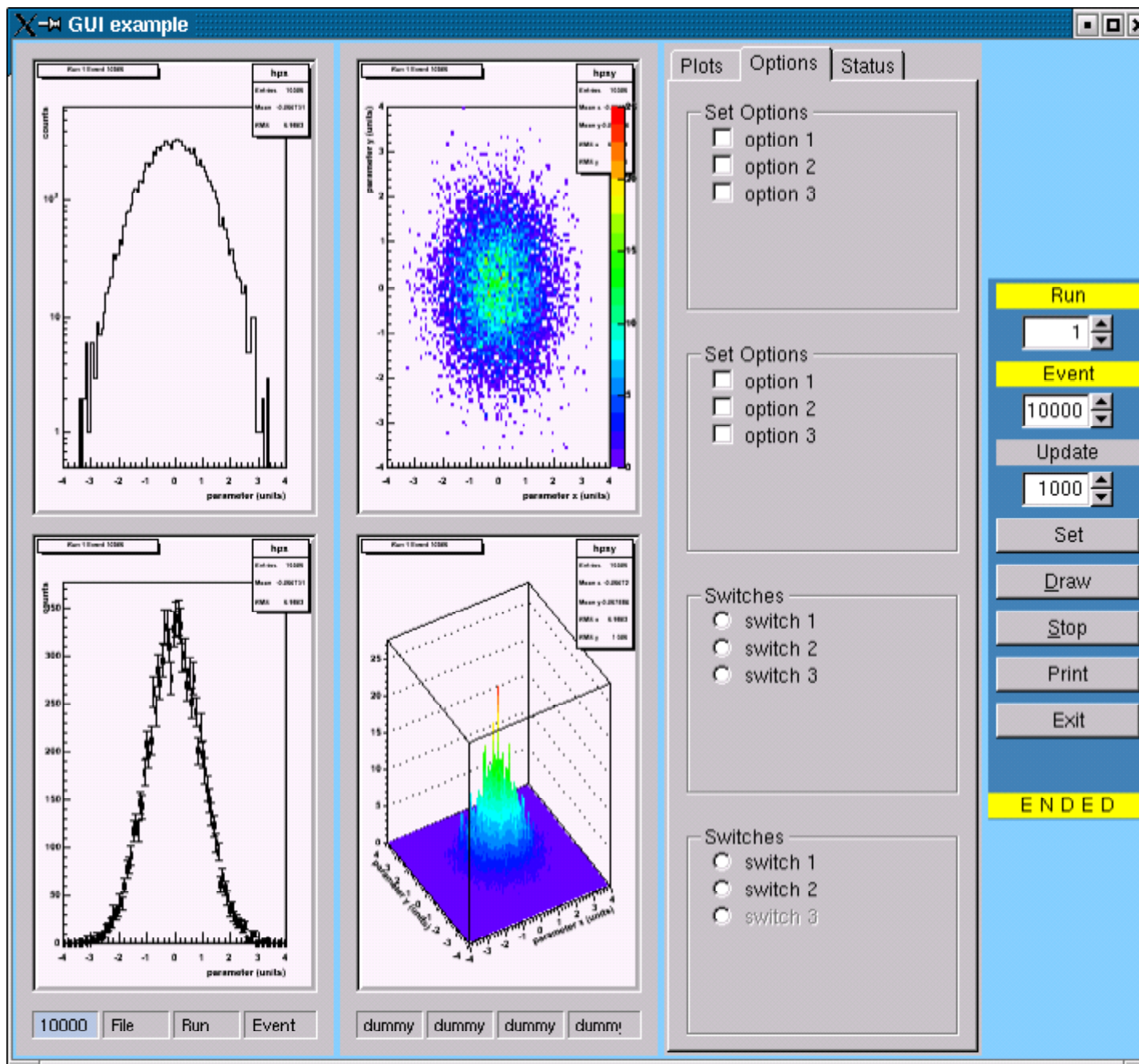
Graphical User Interface

- ▶ . cross-platform GUI application under development for
 - ▷ online histogramming
 - ▷ basic analysis
 - ▷ monitoring
 - ▷ event display
- ▶ . using ROOT GUI classes based on Xclass'95 widget library
(<http://xclass.sourceforge.net>)
- ▶ . basic layout has been implemented and tested on Redhat 7.3 (gcc 3.2.3 or 2.96) with ROOT 4.00 or 3.10

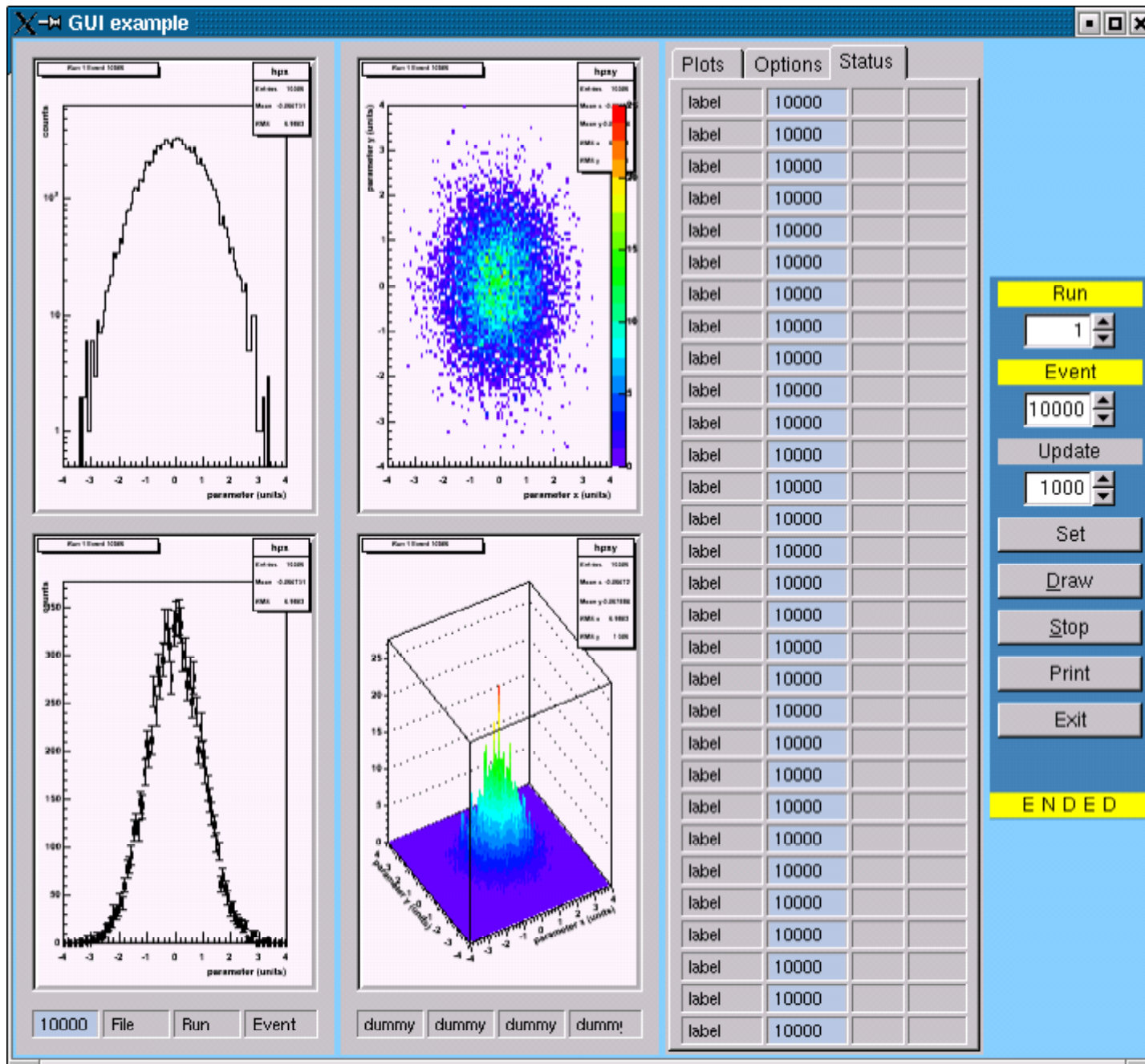
GUI: histograms and action widgets



GUI: tabs for expanded info, settings



GUI: status table



GUI: active pads to invoke ROOT actions

The screenshot shows a ROOT GUI window titled "GUI example" with several active pads and a control panel on the right.

Plots:

- Top Left:** A 2D histogram plot titled "TH1F:hpx" showing counts vs parameter x (units). A context menu is open over it with options: Add, Divide, Draw/Panel, **Fill**, FitPane, Multiply, Rebin, SetMaximum, SetMinimum, Smooth, SetName, SetTitle, Delete, Draw/Class, Draw/Clone, Dump, Inproc, SetDrawOption, SetLineAttributes, SetFillAttributes, SetMarkerAttributes.
- Top Right:** A 2D histogram plot titled "TH2F:hpxy" showing parameter y (units) vs parameter x (units).
- Bottom Left:** A 2D histogram plot showing counts vs parameter x (units).
- Bottom Right:** A 3D histogram plot titled "TH3F:hpxyz" showing counts vs parameter x (units), parameter y (units), and parameter z (units).

Control Panel (Right):

- Plots:** A table with 20 rows, each containing a "label" and a value of "10000".
- Options:** A table with 20 rows, each containing a "label" and a value of "10000".
- Status:** A table with 20 rows, each containing a "label" and a value of "10000".
- Buttons:** Run, Event, Update, Set, Draw, Stop, Print, Exit.
- Input Fields:** A numeric input field with "1" and a spin box with "10000".
- Output:** A yellow bar displaying "ENDED".

Bottom Panel:

- Buttons: File, Run, Event.
- Labels: dummy, dummy, dummy, dummy.

Summary

- ▶ : work on testbeam related software has started and rapid progress is expected
- ▶ : the running and analysis phases of the testbeam program will be fully supported

Calorimeter clustering with **gNIKI**

general **N**odes **I**nterlaced **K**lustering **I**mplementation

G.Mavromanolakis, University of Cambridge



www.hep.phy.cam.ac.uk/~gmavroma/calice/gNIKI

Clustering with gNIKI

g**eneral** **N**odes **I**nterlaced **K**lustering **I**mplementation

▶ . gNIKI

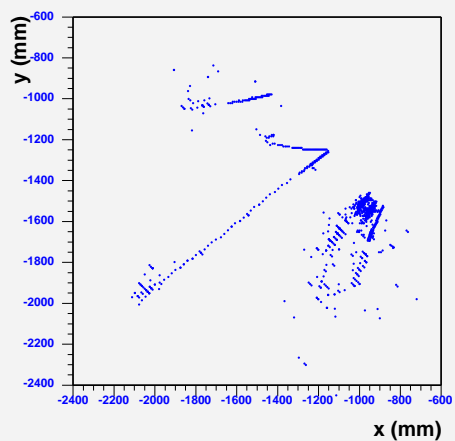
: algorithm based on minimal spanning tree theory to implement a "top-down and then bottom-up" approach to calorimeter clustering

▶ . in brief

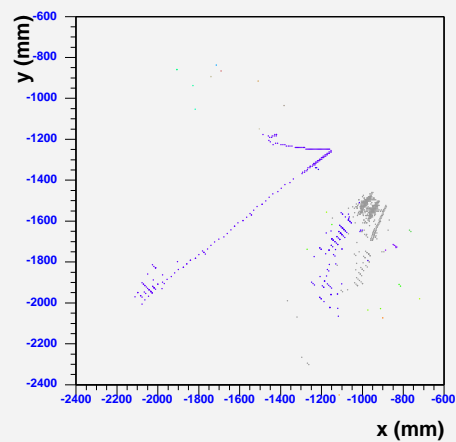
: use MST clustering algorithm with loose cut to perform coarse clustering

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

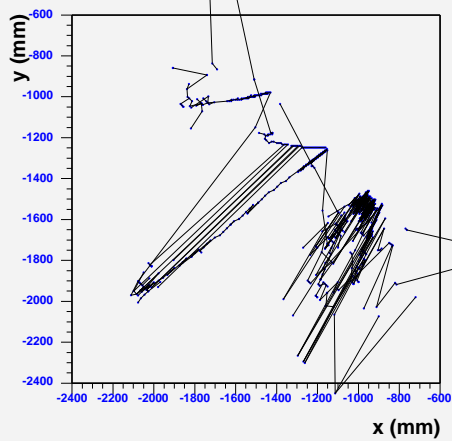
hits



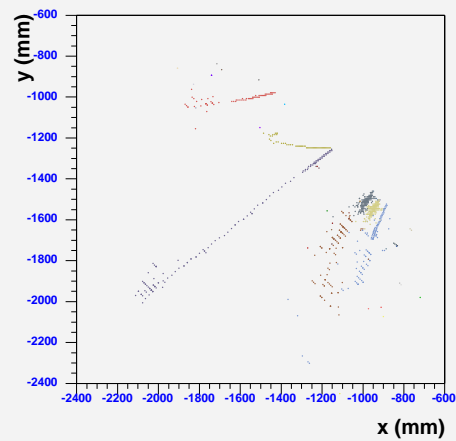
cluster MST



MST



recluster



Calorimeter clustering with **gNIKI**

general **N**odes **I**nterlaced **K**lustering **I**mplementation

G. Mavromanolakis *

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

Contents

1	Introduction	2
2	Graph theory and clustering	2
2.1	2
3	Calorimeter clustering with gNIKI	2
3.1	2
3.2	2
4	Technical description	2
4.1	General Layout	2
4.2	Application Program Interface	2
4.3	Examples	2
5	Summary	2
	References	2

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

draft only

API

```

//.....
// code : gMKI (general Nodes Interlaced Clustering Implementation)
// version : 0.0
// author : G.Mavrounlakis
//.....
#ifdef class_MST
#define class_MST
#endif
#include "MSTclusters.h"
#endif

//.....
class MST
{
private:
    int theM; //
    int *a_mat; //assignment
    int *b_mat; //connectivity
    double *c_mat; //point to point distance
    double *d_mat; //distance matrix

public:
    MST(int);
    ~MST();

    int GetNbOfNodes();
    const int* Read_a_mat();
    const int* Read_b_mat();
    const double* Read_c_mat();
    double* Get_d_mat();
    double* GetDistanceMatrix();
    void PassDistanceMatrix(double*);
    void FillDistanceMatrix();
    void ConstructMST();
    MSTclusters* MergeClusters(float);
    float GetNodeConnectedTo(int);
    float GetMSTLength();
    void Dump();
    static int counter;
};

//.....
#endif

// DESCRIPTION -- INVENTORY
//.....
// gives a set of nodes and a distance matrix the correspond
// spanning tree is constructed by the Prim's method. Then a
// cluster analysis can be performed to merge the clusters.
// MSTclusters.
// thMw : the number of points
// a_mat : integer array with bounds [2*N], a_mat[i]=
// already assigned to the tree, or 0 otherwise
// b_mat : integer array with bounds [2*N], b_mat[i] =
// index of a point to which i is joined.
// c_mat : real array with bounds [2*N], c_mat[i] is the
// distance between points i and b_mat[i].
// d_mat : the lower triangular distance matrix with bounds
// [1:N][1:2].
// MST() : default constructor
// MST(int) : constructor, input is thMw
// ~MST() : destructor
// GetNbOfNodes(): read thMw
// Read_a_mat(): read array a_mat
// Read_b_mat(): read array b_mat
// Read_c_mat(): read array c_mat
// Get_d_mat(): get array d_mat
// GetDistanceMatrix(): same as Get_d_mat()
// FillDistanceMatrix(): example how to create the distance matrix
// GetNodeConnectedTo(int): get the point to which a point i
// is joined.
// ConstructMST(): computes the minimal spanning tree of a d.
// MergeClusters(float): use the minimal spanning tree to do
// linkage cluster analysis, input is the threshold
// GetMSTLength(): return the total length of the mst
// Dump() : print out arrays a_mat, b_mat, c_mat
// counter : static counter to check that all objects are
// deleted.

```

API

```

//.....
// code : gMKI (general Nodes Interlaced Clustering Implementation)
// version : 0.0
// author : G.Mavrounlakis
//.....
#ifdef class_MSTclusters
#define class_MSTclusters
#endif
#include "MSTclusters.h"
#endif

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

private:
    int theM;
    float theCutLevel;
    int *a_mat;
    int *b_mat;
    int *c_mat;
    int* Get_a_mat();
    int* Get_b_mat();
    int* Get_c_mat();

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

    int GetNbOfNodes();
    float GetCutLevel();
    const int* Read_a_mat();
    const int* Read_b_mat();
    const int* Read_c_mat();
    const int* Get_a_mat();
    const int* Get_b_mat();
    const int* Get_c_mat();
    int GetNode(int);
    int GetClusterID(int);
    int GetCluster(int);
    void Dump();
    static int counter;
};

//.....
#endif

// DESCRIPTION -- INVENTORY
//.....
// Class to hold objects created by MST::MergeClusters. One also
// holds the distance out to merge clusters
// thMw : the number of points
// theCutLevel : the distance out to merge clusters
// a_mat : integer array with the point index
// b_mat : integer array with bounds [2*N], b_mat[i] =
// index of a point to which i belongs to the same cluster
// c_mat : integer array with the cluster id that point i was
// joined to
// Get_a_mat(): get array a_mat
// Get_b_mat(): get array b_mat
// Get_c_mat(): get array c_mat
// MSTclusters(int, float): constructor, input thMw of nodes and the
// cut level
// ~MSTclusters(): destructor
// GetNbOfNodes(): read thMw
// GetCutLevel(): read theCutLevel
// Read_a_mat(): read array a_mat
// Read_b_mat(): read array b_mat
// Read_c_mat(): read array c_mat
// Get_a_mat(): same as Read_a_mat()
// Get_b_mat(): same as Read_b_mat()
// Get_c_mat(): same as Read_c_mat()
// GetClusterID(int): input is i, returns b_mat[i]
// GetCluster(int): read number of clusters found by single linkage
// clustering
// Dump() : print out arrays a_mat, b_mat, c_mat
// counter : static counter to check that all objects are
// deleted.

```

example

```

//.....
// code : gMKI (general Nodes Interlaced Clustering Implementation)
// version : 0.0
// author : G.Mavrounlakis
//.....
#ifdef class_MSTclusters
#define class_MSTclusters
#endif
#include "MSTclusters.h"
#endif

//.....
int main()
{
//Open profile with roottree
FILE *f = new FILE(input_root);
//Create some histogram
//Get/Set branches of roottree

//Delete objects
delete tMST; // tMST=0;
delete tMSTclusters; tMSTclusters=0;
delete theEventCluster; theEventCluster=0;

//end loop over events

// Save histo
return 1;
}

//end main()

```

```

float level = out->GetMSTCut();
MSTclusters *tMST=new MSTclusters(level);

//Loop over points of this event and fill theEventCluster
TCluster *theEventCluster = new TCluster(theM);
theEventCluster->InitCount();
//the "Node" = theEventCluster->GetListOfNodes();

//Loop over MSTclusters of this event and fill theEventCluster
for(int i=0; i<tMSTclusters->GetN(); i++)
{
TCluster *tEventMSTCluster = theEventCluster->GetMSTCluster(i);
theEventCluster->UpdateWithDaughter(tEventMSTCluster);
}

//Do analysis, fill histo

//Delete objects
delete tMST; // tMST=0;
delete tMSTclusters; tMSTclusters=0;
delete theEventCluster; theEventCluster=0;

//end loop over events

// Save histo
return 1;
}

//end main()

```

gNIKI

▶ . release version 0.0

- : standalone C++ code arranged in 6 classes
(programmatically not the best code you've ever seen but it works)

- : get source from

 - www.hep.phy.cam.ac.uk/~gmavroma/calice/gNIKI

- : example applications on howto link and read data from
ROOT or LCIO included

▶ .

- : write-up as LC Note in preparation (hopefully available soon),
in the meantime a list of talks and documents are on the webpage