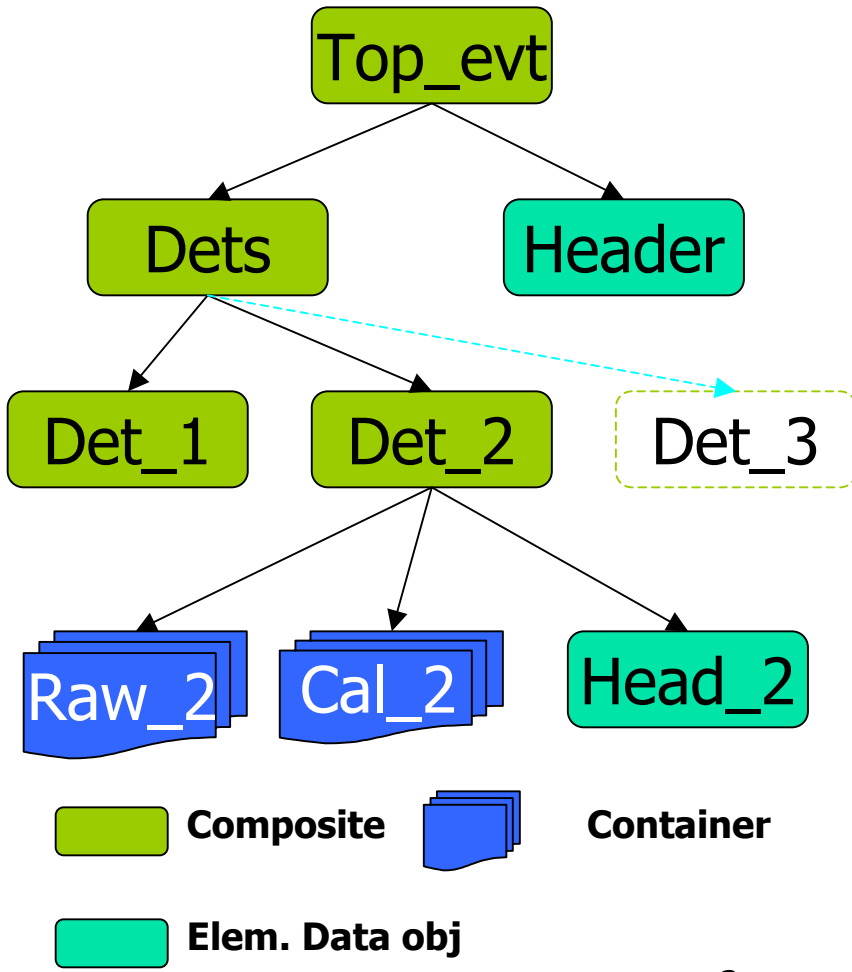
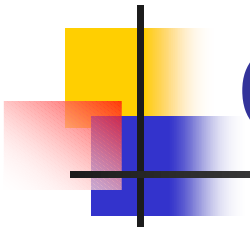
The GO4 Composite Event Structure

- Motivations
  - The Event Structure
  - Goals
  - Requirements
- OO model implementation
- User Interface
  - Modeling an event structure
  - Data retrieval
- Go4 examples (FOPI)

The Event Structure

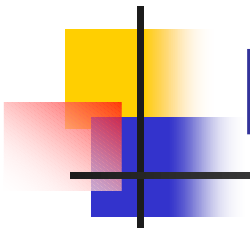
- Structural organization of data record after primary interaction.
- Should represents the natural experimental setup (geometry, detectors, …)
- Fast direct access of components (indexation needed)
- Full or Partial IO: need to map the event object into a ROOT TTree
- General design to fit to different needs





Goals

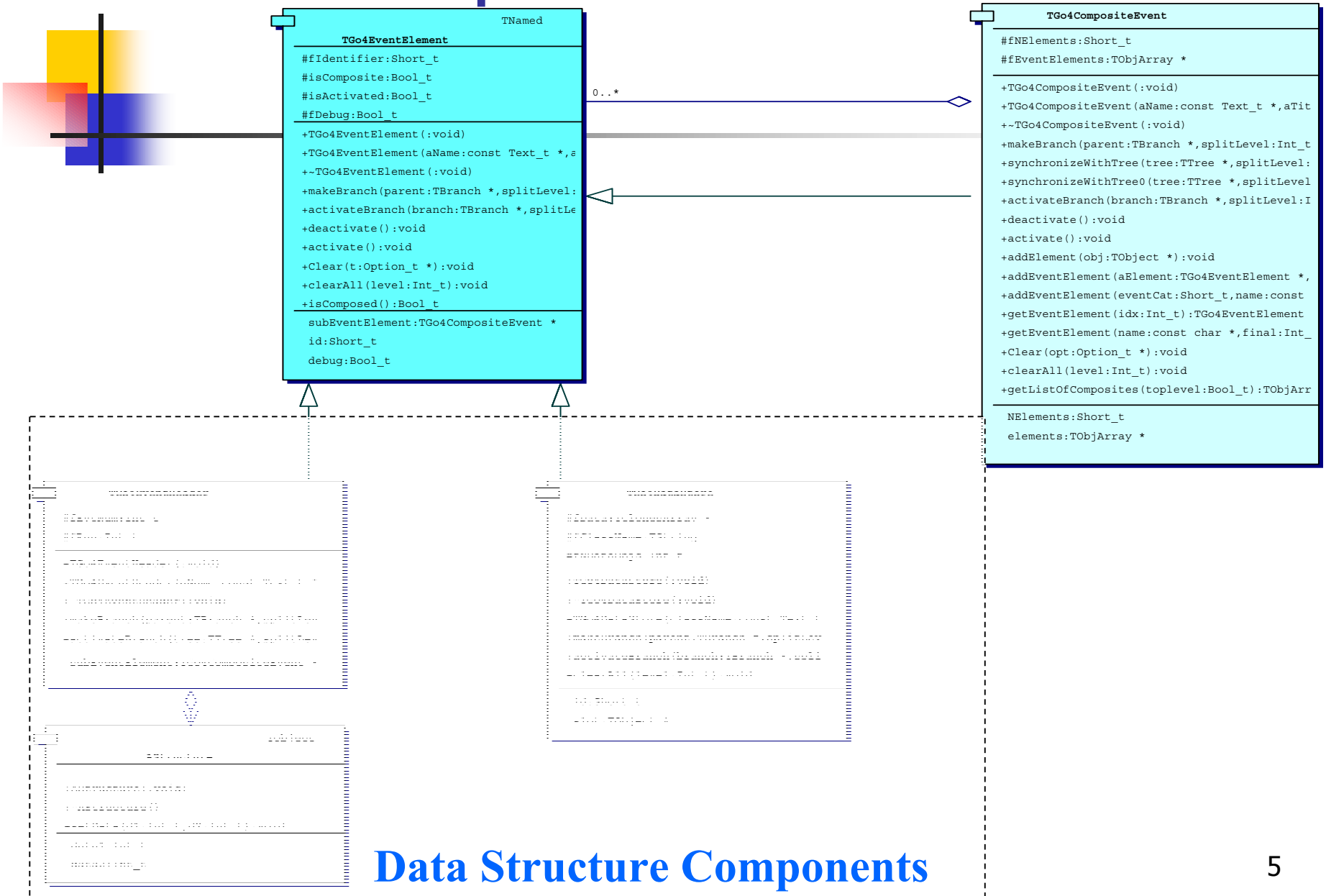
- Let user build complex Event Structure out of simple Data Object components
- Treats single Data Objects and composition of Data Objects uniformly
- User can group components to form larger components
- Easy data retrieval from file



Requirements

- Use pure ROOT class for IO mechanism
  - No changes in TTree or TBranchElement
- Recursive method to synchronize the composite structure with TTree
  - To retrieve user Data, no need for :
    - TTree::MakeCode()
    - TTree::MakeClass()

OO composite model

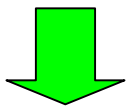


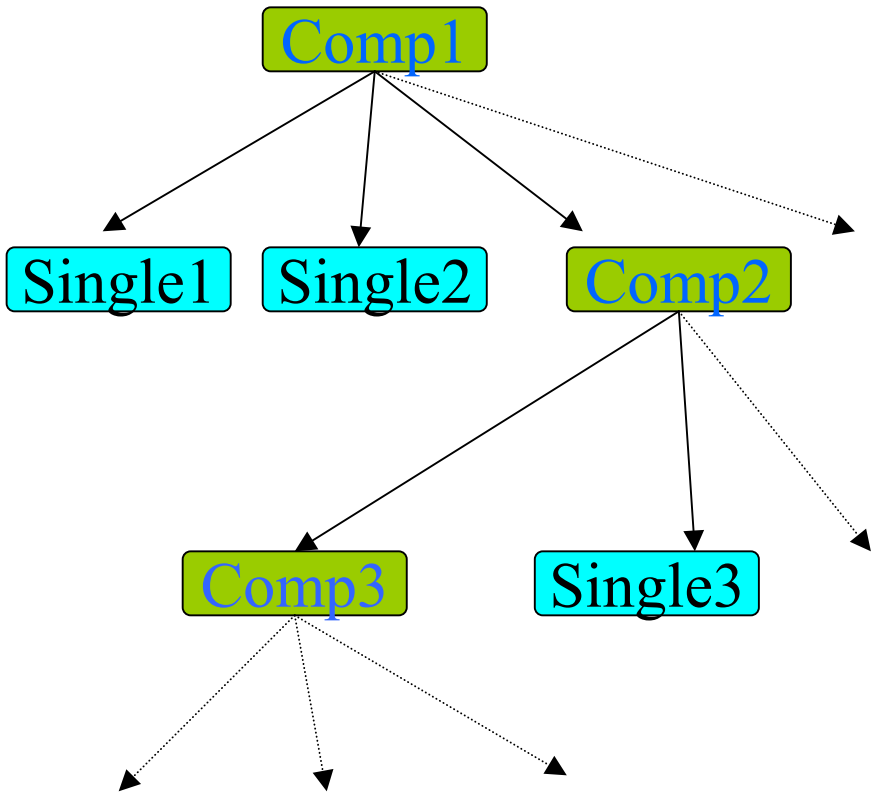
Data Structure Components

Synchronizing with TTree

Use of recursive algorithms

(composite OO model)





- Create branches in TTree:

TGo4EventElement::makeBranch()

TGo4CompositeEvent::makeBranch()

- Synchronize objects with TTree:

TGo4CompositeEvent::Synchronize(TTree*T)

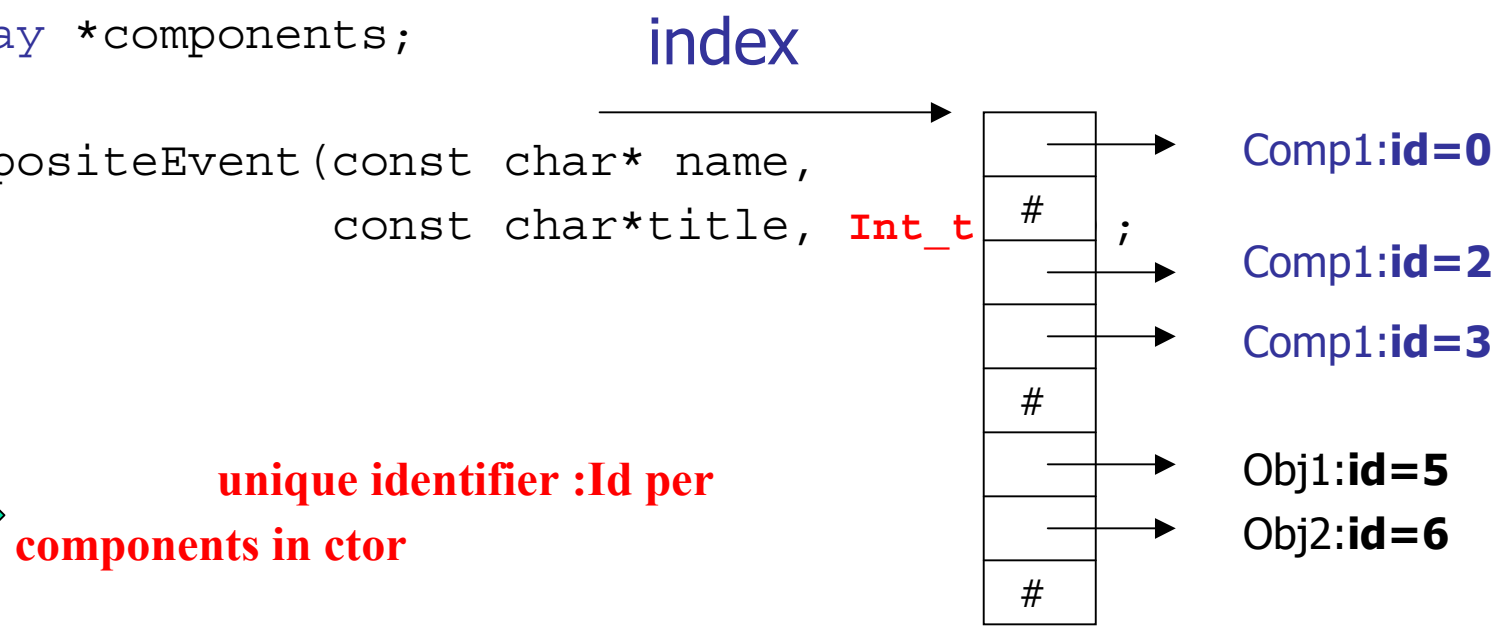
- Partial IO:

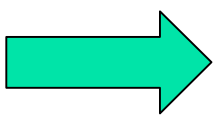
TGo4CompositeEvent::activate();

TGo4CompositeEvent::deactivate();

Creating components

```
Class TGo4CompositeEvent: public TGo4EventElement{
private:
TObjArray *components;
public:
TGo4CompositeEvent(const char* name,
                   const char*title, Int_t ;
...
};
```





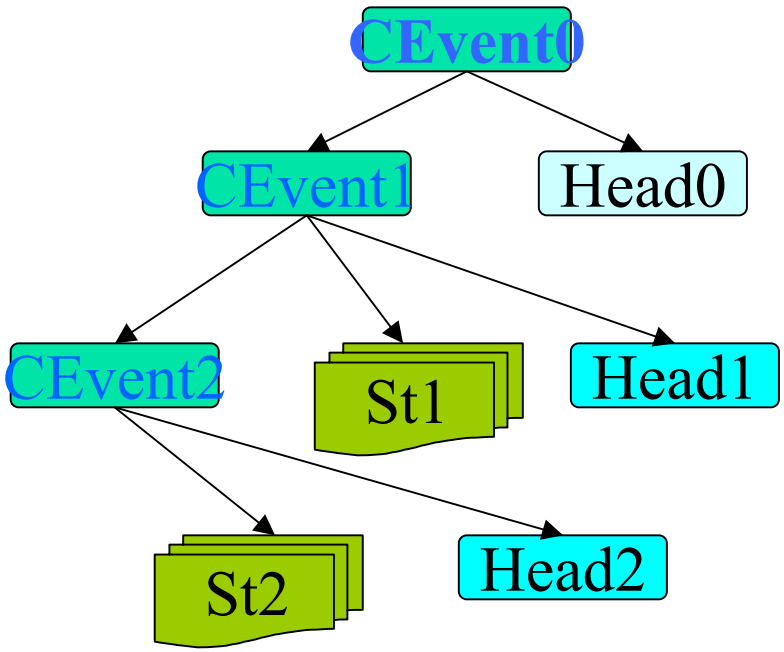
unique identifier :Id per components in ctor

Creating components

```
{ //user interface example
TTree *tree = new TTree("T","test");
// create some composites
TGo4CompositeEvent* event0,*event1,*event2;
CEvent0= new TGo4CompositeEvent("cEvent0","top",id1);
CEvent1= ...
// create some containers for data storage
TGo4ClonesElement* st0,*st1,*st2;
st0 = new TGo4ClonesElement("Htrack",100,"Det1",idc1);
...
// create some headers
TGo4EventHeader *head0,*head1,*head2;
head0 = new TGo4EventHeader("header0","header for
   det0",idh1);
...
}
```

Modeling an event

```
{//Event structure modeling
// 2th composite
CEvent2->addElement(head2);
CEvent2->addElement(st2);
//first composite
CEvent1->addElement(head1);
CEvent1->addElement(st1);
CEvent1->addElement(CEvent2);
// add in top composite event
CEvent0->addElement(head0);
CEvent0->addElement(CEvent1);
// set top level branch
TBranch*b = T->Branch("top",
"CompositeEvent",&CEvent0);
//compose tree structure
CEvent0->makeBranch(b,split);
}
```



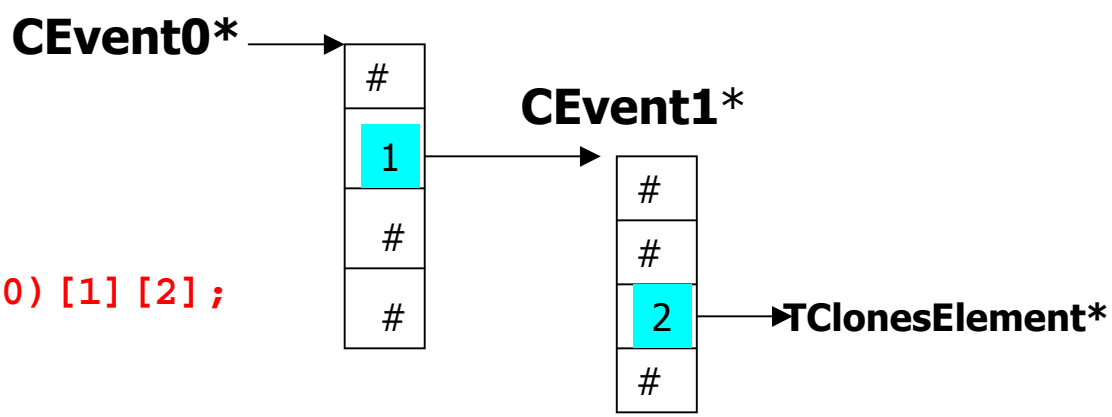
Data retrieval (1)

```
{
TFile *f = new TFile("combo.root");
TTree *tree= (TTree*) f->Get("T");
// create an instance of composite event
TGo4CompositeEvent* cEvent0 = new TGo4CompositeEvent()
// Use of a recursive algorithm to recreate
// the composite event in memory from the
// branches in TTree
CEvent0->synchronizeWithTree(T,split);
//Retrieving sub-elements
TGo4CompositeEvent* c1 = CEvent0->getElement("cEvent1");
TGo4Element *e2 = c1->getElement("head1");
CEvent0->getElement("cEvent2")->deactivate();
// then do event loop
}
```

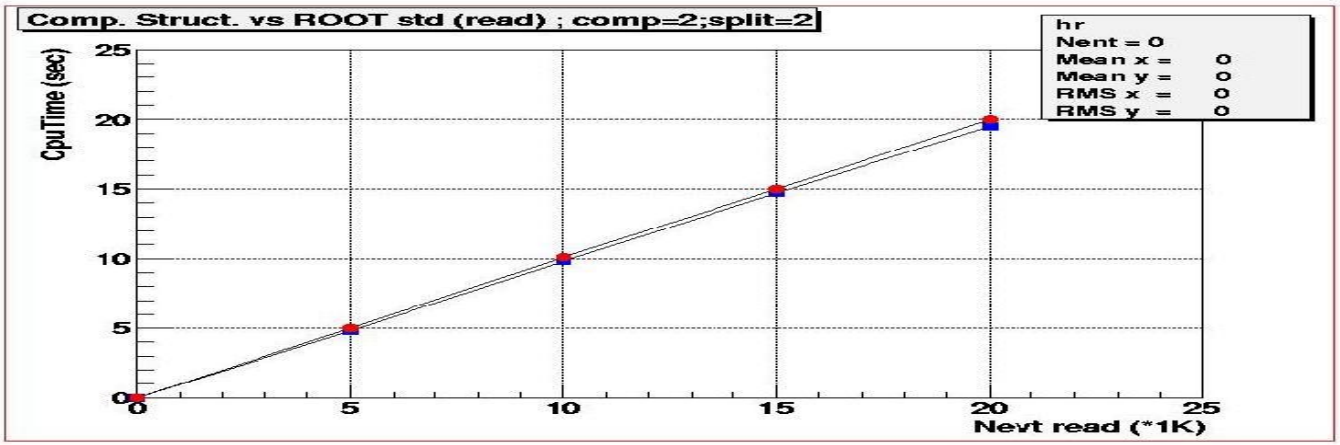
Data retrieval (2)

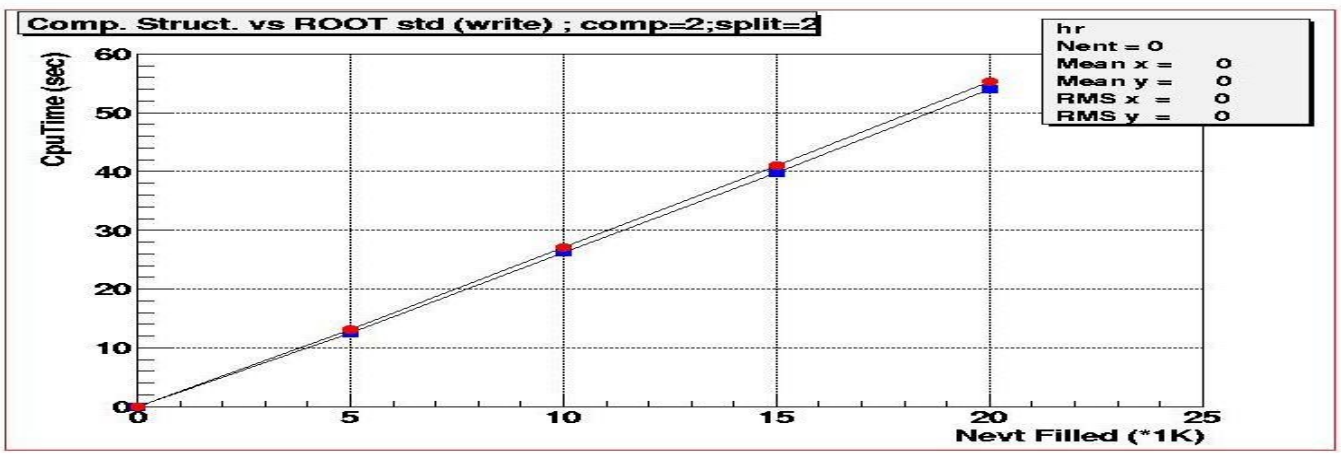
```
{
//Retrieving component by name
TGo4CompositeEvent* c1 = CEvent0->getElement("cEvent1");
TGo4Element *e2 = c1->getElement("head1");
//deactive IO for this component
CEvent0->getElement("cEvent2")->deactivate();
```

```
//Direct access by indexes
TClonesElement* cl =
(TClonesElement*) &(*CEvent0)[1][2];
}
```

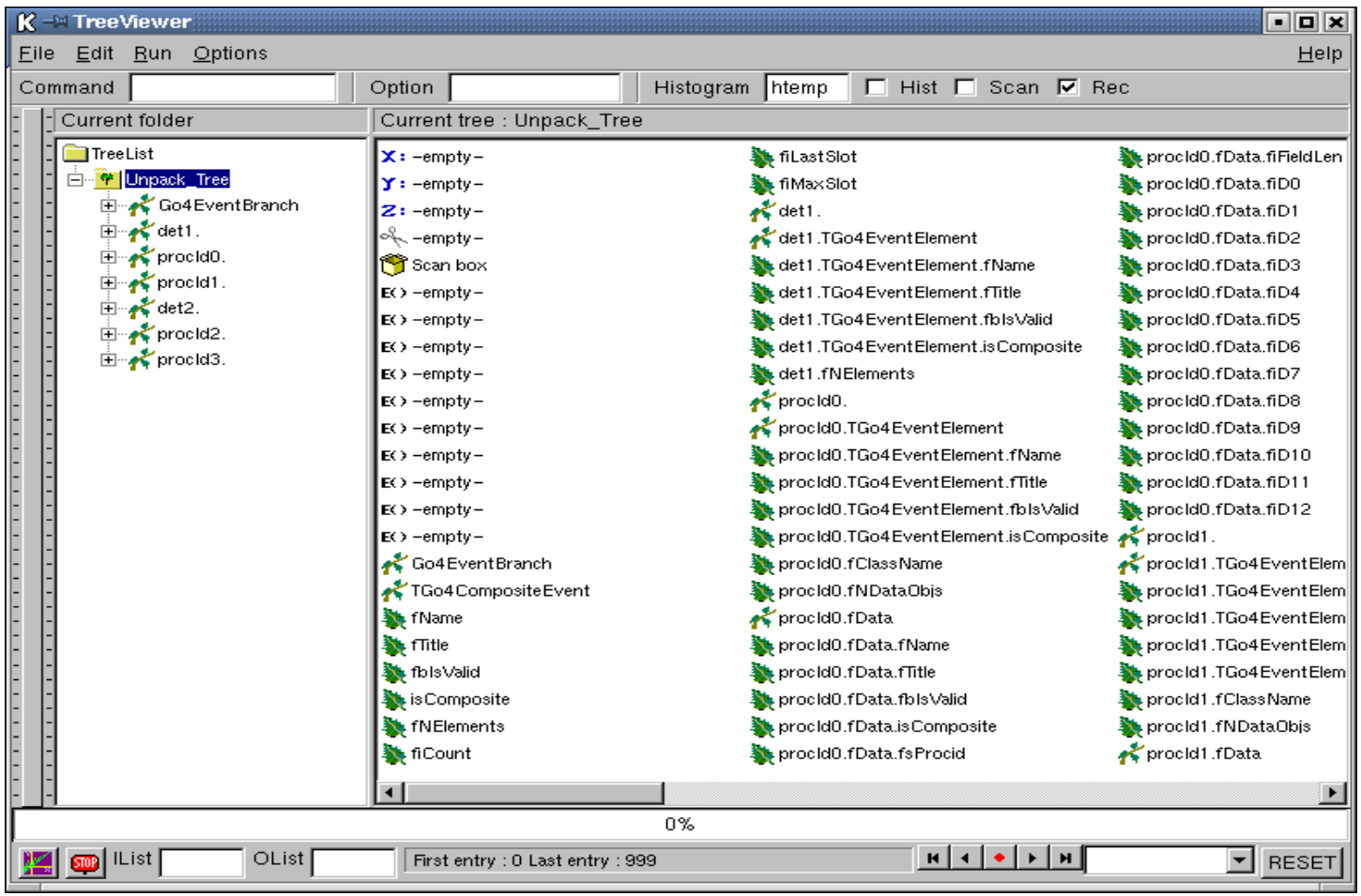


Benchmarks

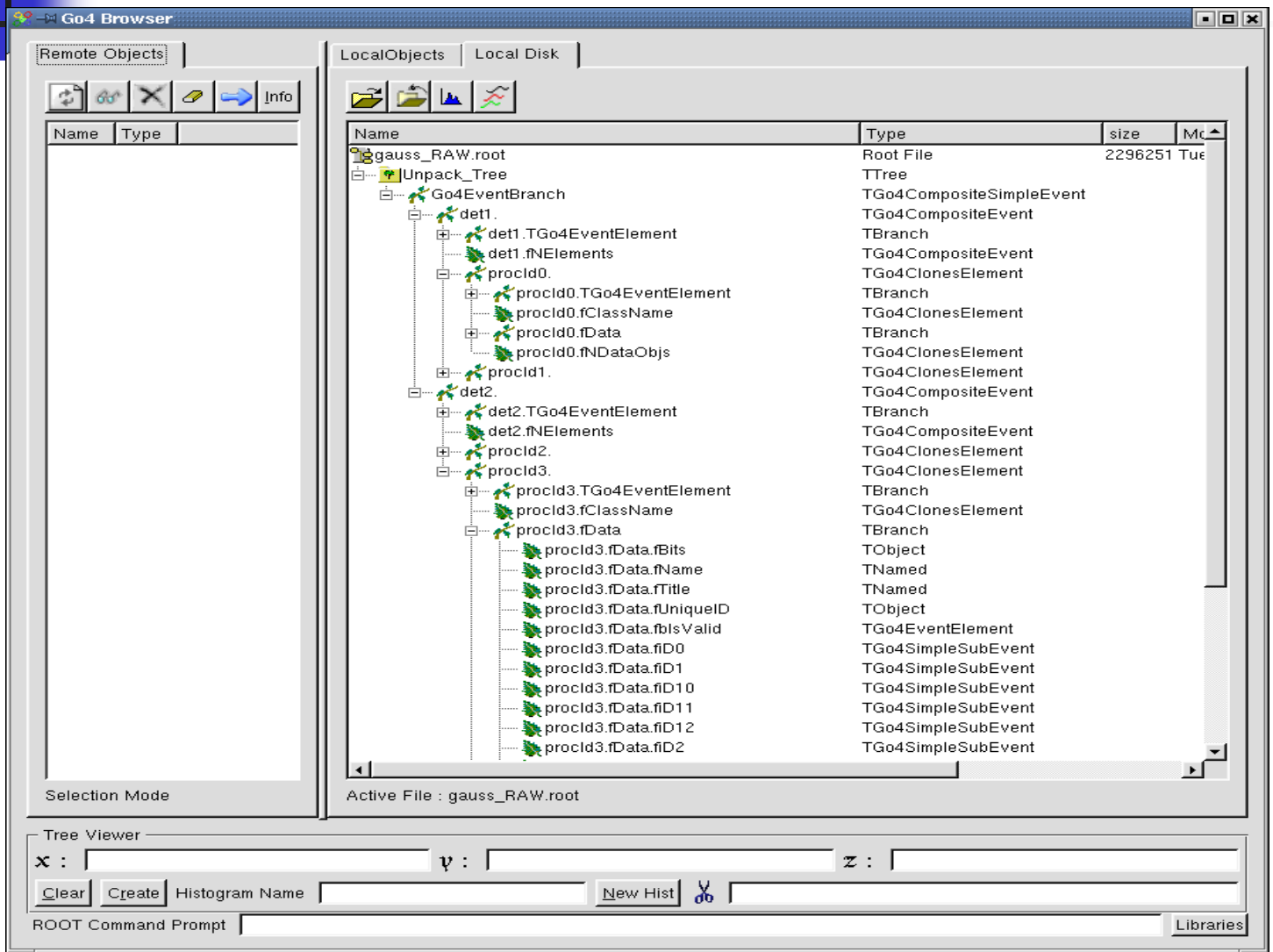

Comp. Struct. vs ROOT std (read) ; comp=2;split=2
hr
Nent = 0
Mean x = 0
Mean y = 0
RMS x = 0
RMS y = 0
CpuTime (sec)
0
5
10
15
20
25
Nevt read (*1K)


Comp. Struct. vs ROOT std (write) ; comp=2;split=2
hr
Nent = 0
Mean x = 0
Mean y = 0
RMS x = 0
RMS y = 0
CpuTime (sec)
0
10
20
30
40
50
60
0
5
10
15
20
25
Nevt Filled (*1K)

Go4Browser vs ROOT



Go4Browser vs ROOT



Go4 examples (FOPI)

- Install FOPI example
- Problem statement
  - Raw data on MBS level
  - Which event structure (3 variants)?
- Create the composite events
- How to fill the event
  - Unpack+ Calibrate step
- How to read the event
  - Go4Browser
  - ROOT compiled program (t4pread)

Go4 examples (FOPI)

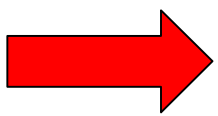
- copy the tar-ball to your $HOME dir:
  - `mkdir Go4FopiExample;`
  - `cd Go4FopiExample;`
  - `cp /misc/denis/go4_fopi.tar.gz .`

- Unpack it:

```
> tar xvfz go4_fopi.tar.gz
> ln -s /s/dbertini/test4p.lmd test4p.lmd
> make all (create MainUserAnalysis)
> make reader (create t4pread)
```

Problem statement

- Raw Data format on MBS level
  - Fastus ID :[1,2,3,...]
  - VME board Slot ID :[1-24]
  - TDC-ADC channel ID: [1-96]
  - Channel value

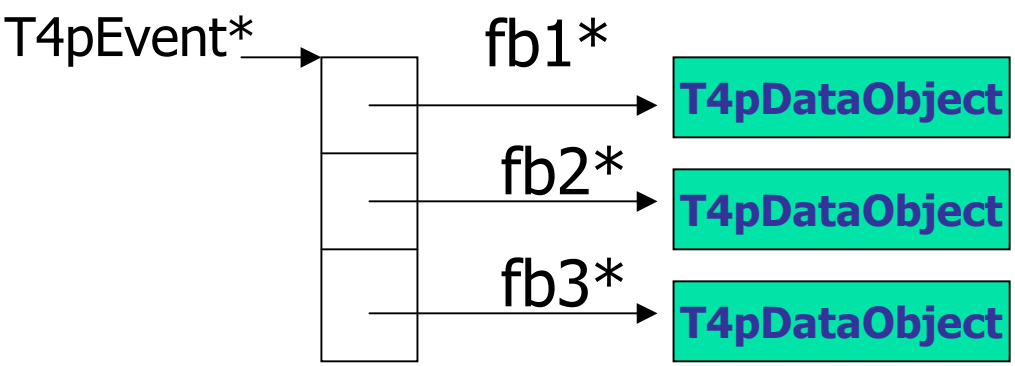


Need to reproduce hardware setup for online monitoring

Example contents

- Base Go4 Classes
  - T4pEventFactory + T4pCalibrateFactory
  - T4pEventProcessor + T4pCalibrateProcessor
  - T4pEvent (event structure)
- Fopi Data Object definition
  - T4pDataObject.h
    - Class T4pClonedDataObject
    - Class T4pDataObject
    - Class T4pSlotDataObject

Event structure (1)



Have a look at :

- T4pDataObject.h:

Class T4pDataObject:
public TGo4EventElement{
protected:
Int_t data[Maxslot][MaxChannel];
...
};

- **T4pEvent.cxx**
- **T4pEvent()** **if(example==1){}**
- **AddF()** **if(example==1){}**

Run the example (1)

- 1_ Use Go4 in batch mode

```
>MainUserAnalysis test4p nevt 1
>ls *.root
   >>> test4p_RAW.root
   >>> test4p_CAL.root
```

- 2_ Use Go4 in remote mode

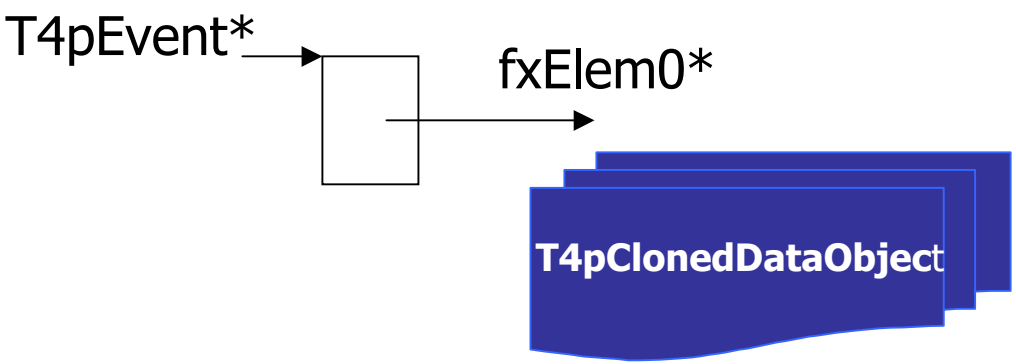
```
>go4gui &
>MainUserAnalysis -gui test4p localhost 5000 1
```

- Then from the GUI:
  - Get Active Config
  - Start

Data organization (1)

- From Go4GUI Viewer (remote mode):
  - Find how to create :
    - 1D-hist of (fastbus=2,slot=24,chan=35)
    - 1D-hist of (fastbus=2,slot=14) all channels
    - 1D-hist of fastbus=2, all slots all channels
- Do the same from file:
  - Load lib: `libGo4CompositeStoreExample.so`
  - Open the root files from the file dialog

Event structure (2)



Have a look at :

- **T4pDataObject.h:**

Class T4pClonedDataObject:
public TGo4EventElement{
protected:
Int_t fbus;
Int_t slot;
Int_t channel
Int_t data;
...
};

- **T4pEvent.cxx**
  - **T4pEvent() :**
    else if(example==2){}
  - **::AddF() elseif(example==2){}**

T4pEvent::AddF() (2)

```
else if (example== 2){
    T4pClonedDataObject *t= NULL;
    t=(T4pClonedDataObject*) fxElem0->getSlot();
            if (t) {
                //only reserved area, need to call ctor object
                  t = new (t) T4pClonedDataObject();
                              t->setFbus(Fnumber);
                              t->setSlot(Slot);
                              t->setChannel(Chan);
                              t->setData(Data);
                            }
   }
```

Run the example (2)

- 1_ Use Go4 in batch mode

```
>MainUserAnalysis test4p.lmd nevt 2
>ls *.root
   >>> test4p_RAW.root
   >>> test4p_CAL.root
```

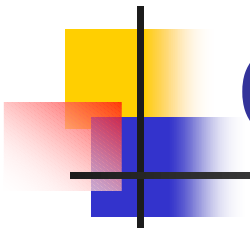
- 2_ Use Go4 in remote mode

```
>go4gui &
>MainUserAnalysis -gui test4p localhost 5000 2
```

- Then from the GUI:
  - Get Active Config
  - Start

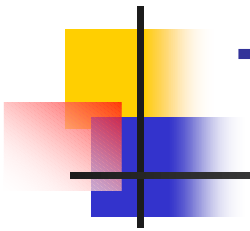
Data organization (2)

- From Go4GUI Viewer (remote mode):
  - Find how to create :
    - 1D-hist of (fastbus=2,slot=24,chan=35)
    - 1D-hist of (fastbus=2,slot=14) all channels
    - 1D-hist of fastbus=2, all slots all channels
- Do the same from file:
  - Load lib: `libGo4CompositeStoreExample.so`
  - Open the root files from the file dialog



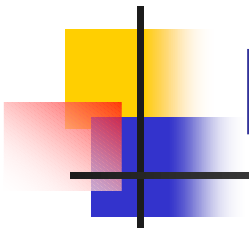
Calibration

- Correct each channel content:
  - Class T4pCalibration
    - gain(fastbus,slot)
    - offset(fastbus,slot)
- Second step :
  - Class `T4pCalibrateProcessor`
  - Data processing:
    - `::buildCalibratedEvent(T4pEvent*)`



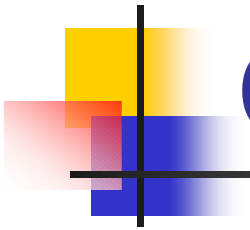
T4pread

- Mini Analysis of output ROOT files containing a Composite Event
- Use synchronizeWithTree() for dynamic creation of the event
- Use an internal eventloop
  - Event structure only overwritten
- Usage: `t4pread <root_filename>`



Performance penalty

- The event structure is the backbone of the analysis
- A wrong design can create performance penalty for **all users**
- Try to compare the performance in Writing/Reading data (memory, cpu, output file size) of
  - example 0 (extended TTree -> 1 branch /slot)
  - example 1
  - example 2



Conclusion

- A composite Model for Event structure has been developed and tested
- No sub-classing of ROOT IO classes needed
- Easy retrieval of data via recursive algorithm
- No limitation in the composition of data objects