Software for the CALICE Project(s)

Roman Pöschl
DESY Hamburg

Applying/Testing ILC Software Tools in ILC Detector Development

ILC Software Workshop DESY June 2005

Introduction

CALICE collaboration is preparing/performing large scale testbeam
(See Calo Session)
Primary sites DESY and FNAL

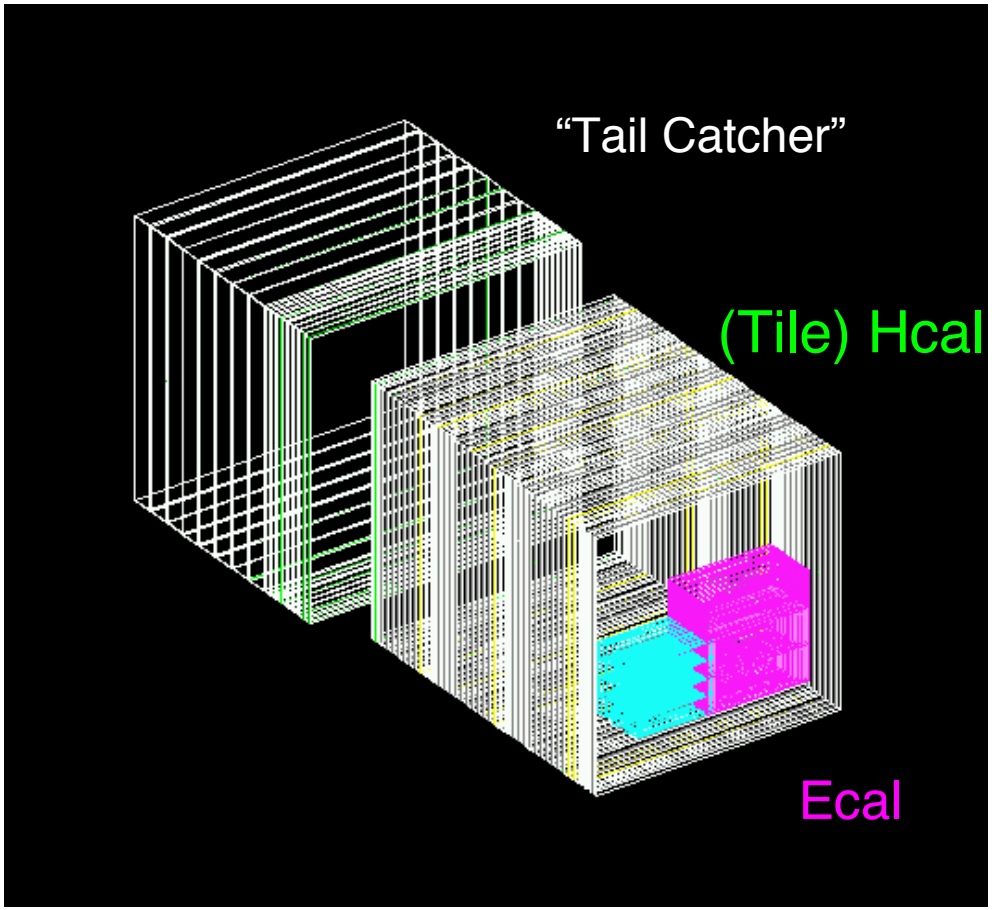
Testbeam program poses software “challenges”

- Detailed simulation of testbeam setup

- Data processing from Raw Data to final Clusters in user friendly way

- Handling of Conditions Data

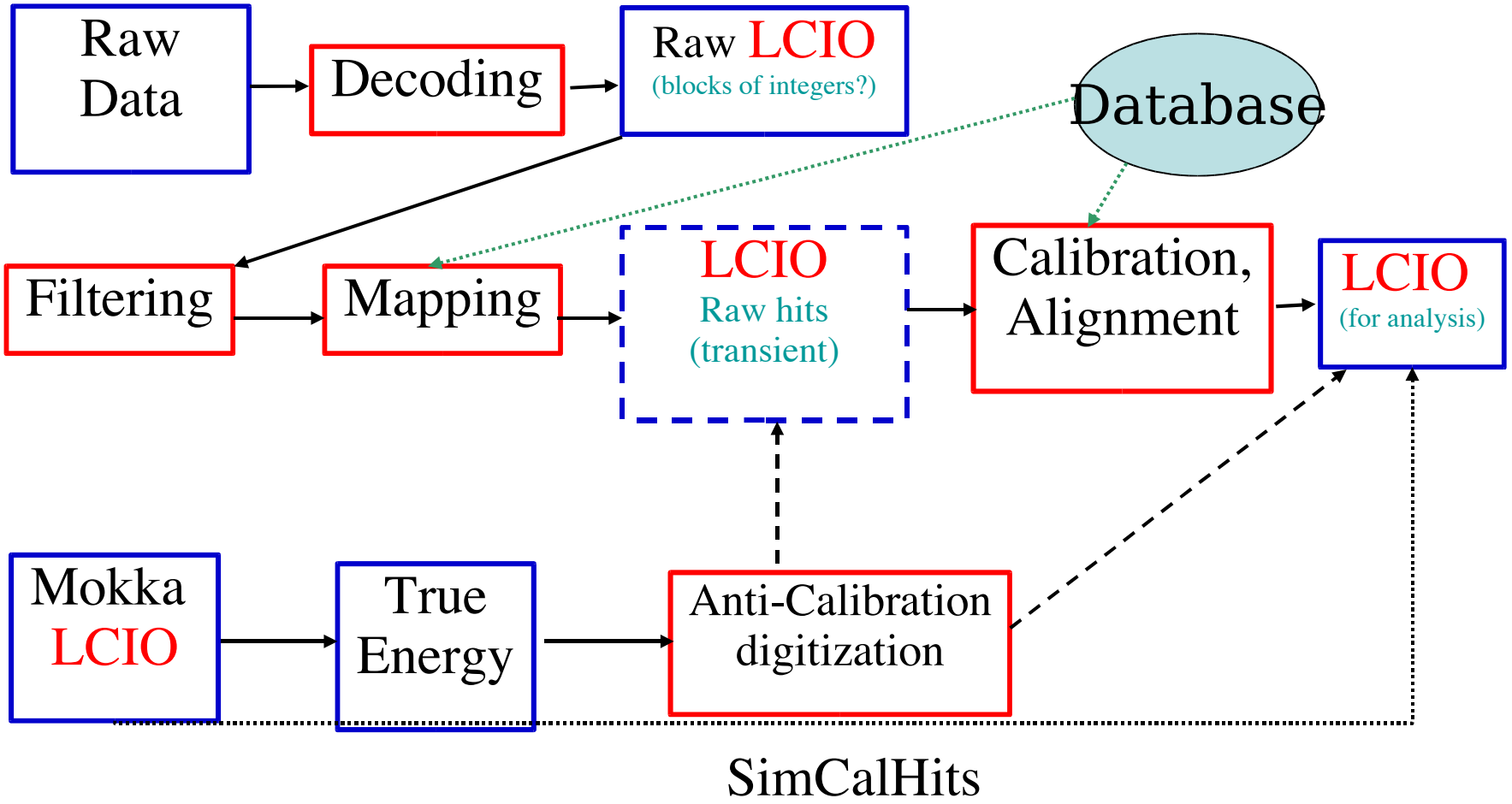
Testbeam software is to be developped within ILC framework



Complete testbeam setup available in Mokka
+ tracking devices (new since 04-02)

Dataflow in CALICE Testbeam

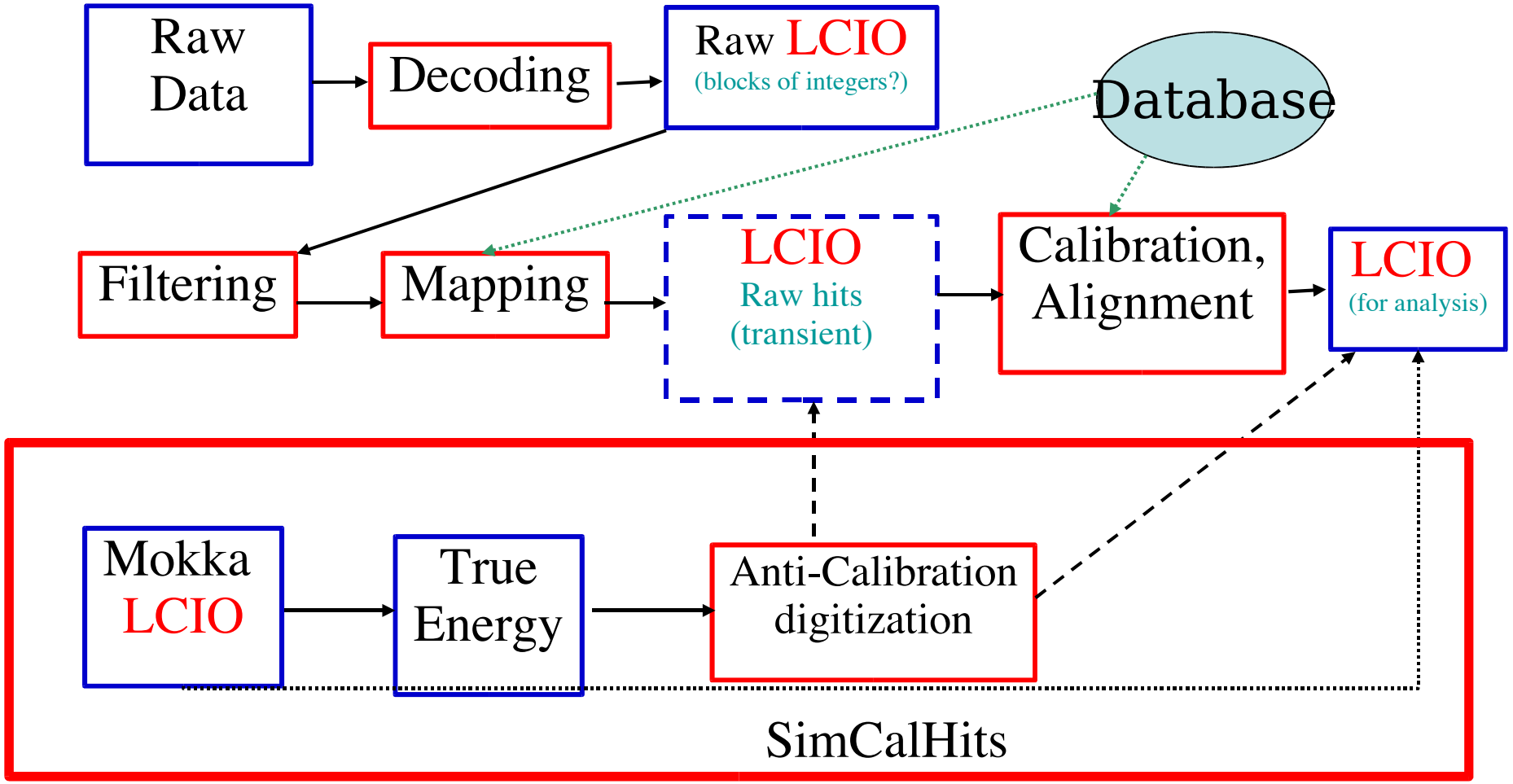
LCIO as backbone of Testbeam Analysis



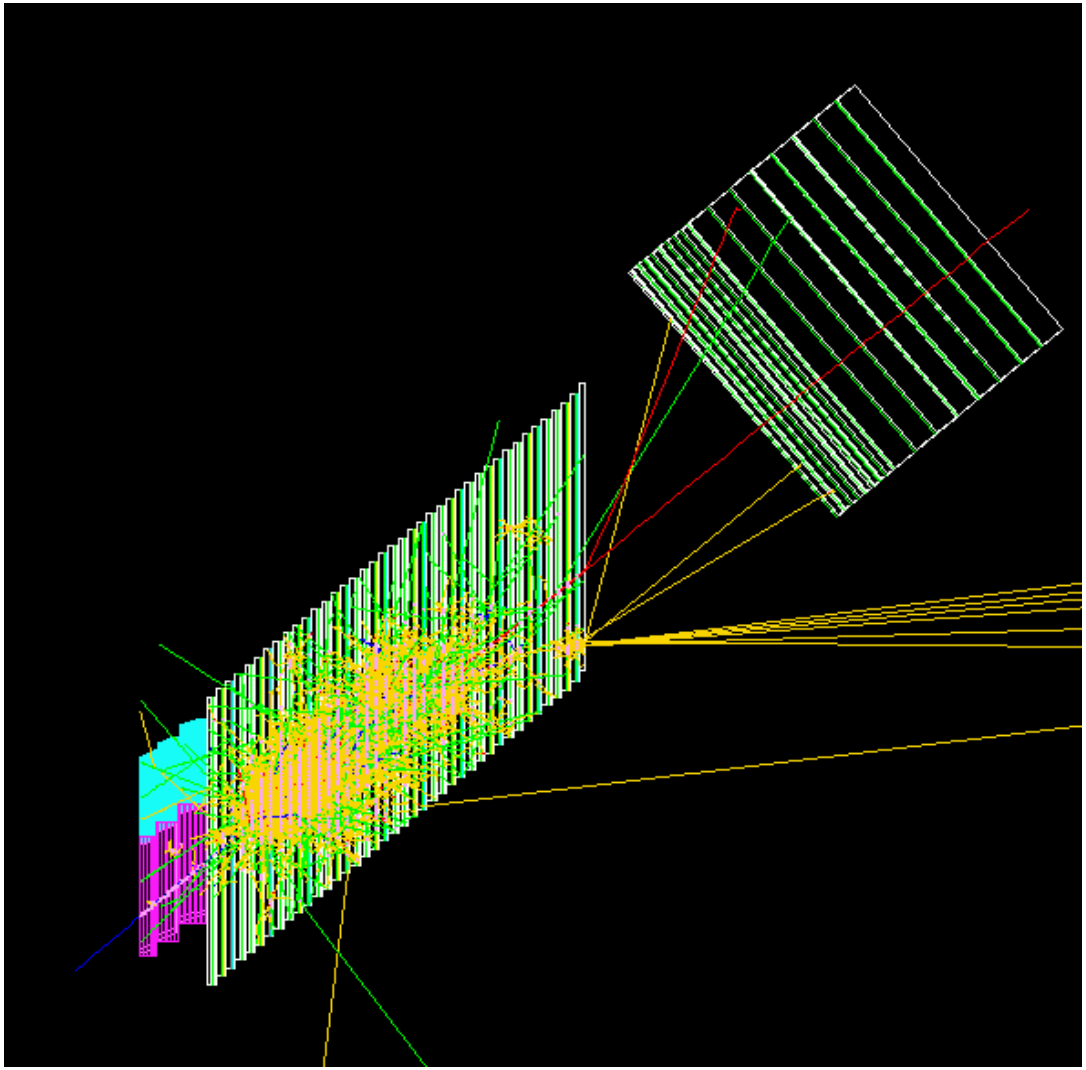
Realization of Scheme ?

Dataflow in CALICE Testbeam

LCIO as backbone of Testbeam Analysis

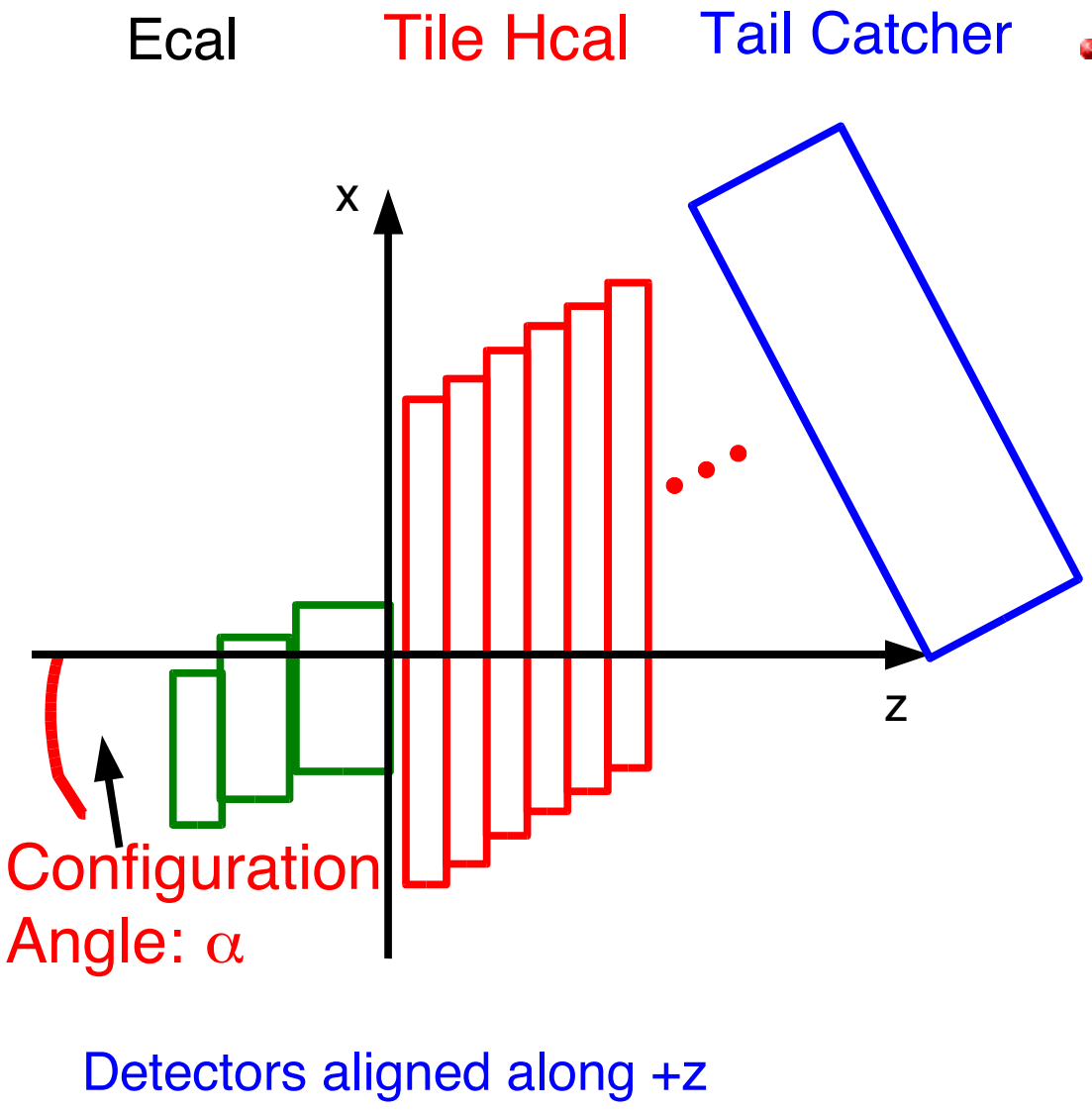


Testbeam Simulation with Mokka



Study of different impact angles

Implementation Issues



Detectors aligned along +z

- Implementation allow for different configuration angles

The layers of the detectors are shifted and the beam is rotated

Implementation is part of Mokka releases since Mokka 03-02

Geometry/tbeam area:
names ...03... (e.g. Tbhcal03.cc)

- Communication of parameters between drivers ?

Since Release 4.0

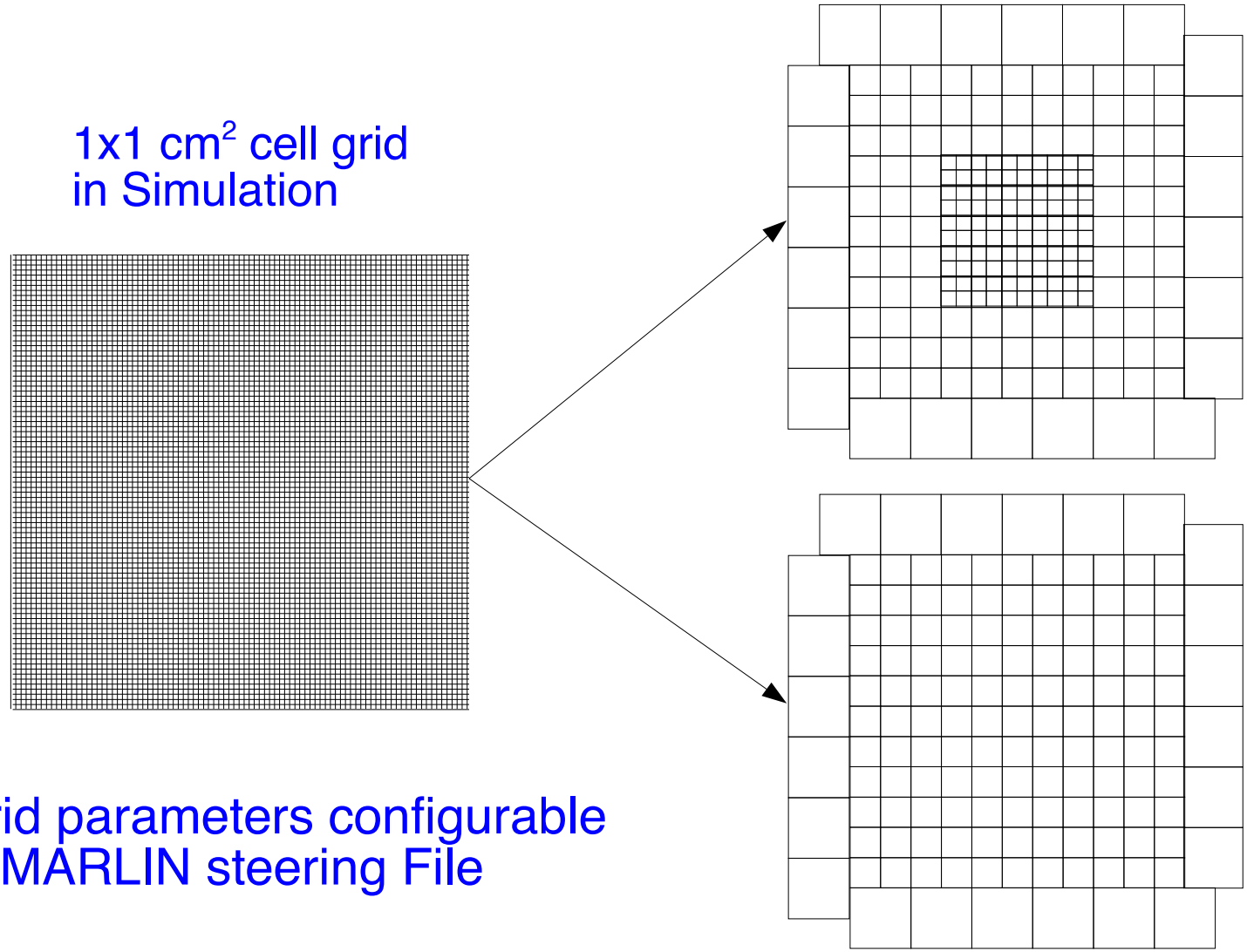
Sharing parameters between drivers
(see talk by Paolo)

MARLIN in CALICE Project – The Ganging Processor

Combines calo cells from simulation output into 'real' cells

Real Cell Grid(s) of Tile Hcal

1x1 cm^2 cell grid
in Simulation

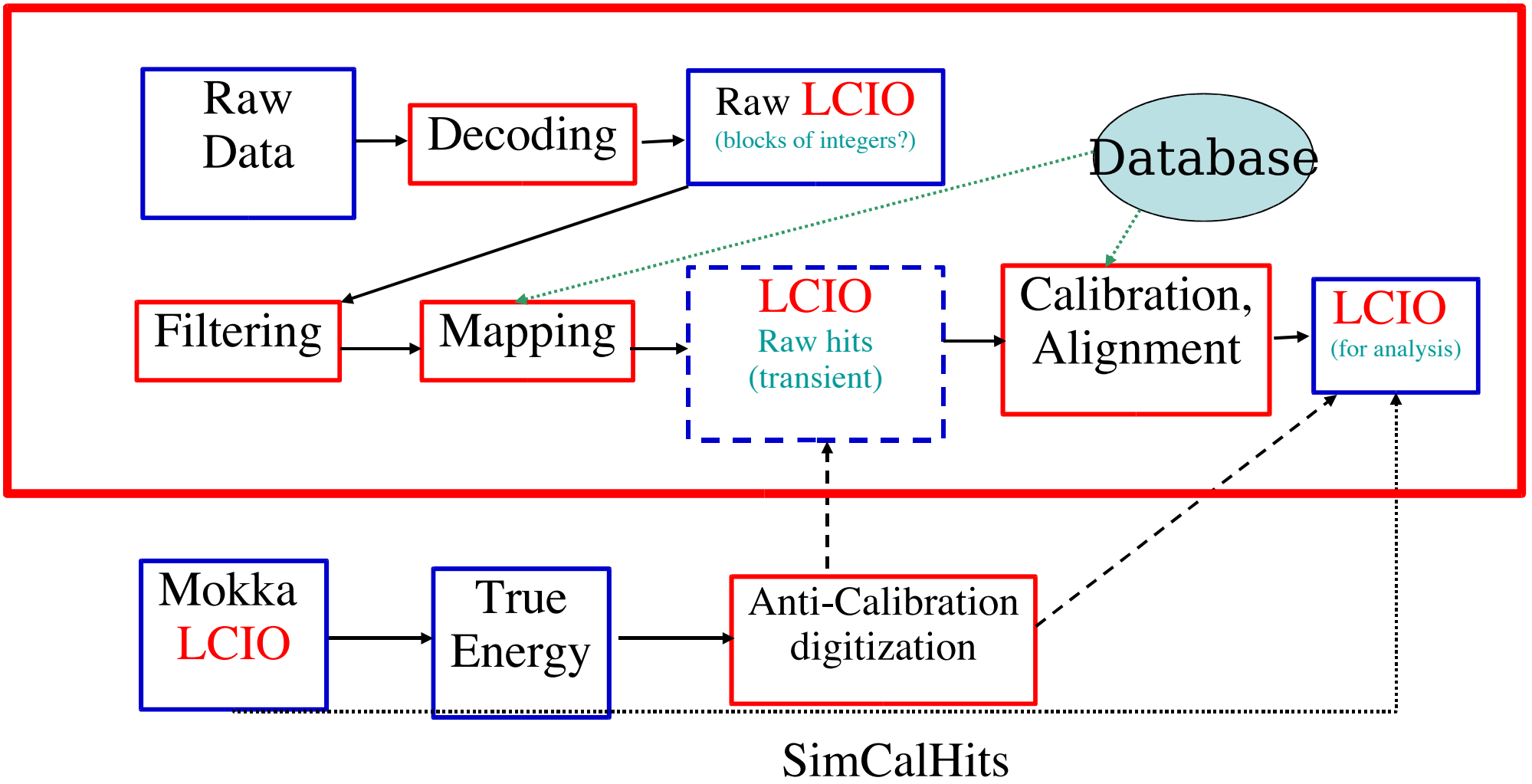


Grid parameters configurable
in MARLIN steering File

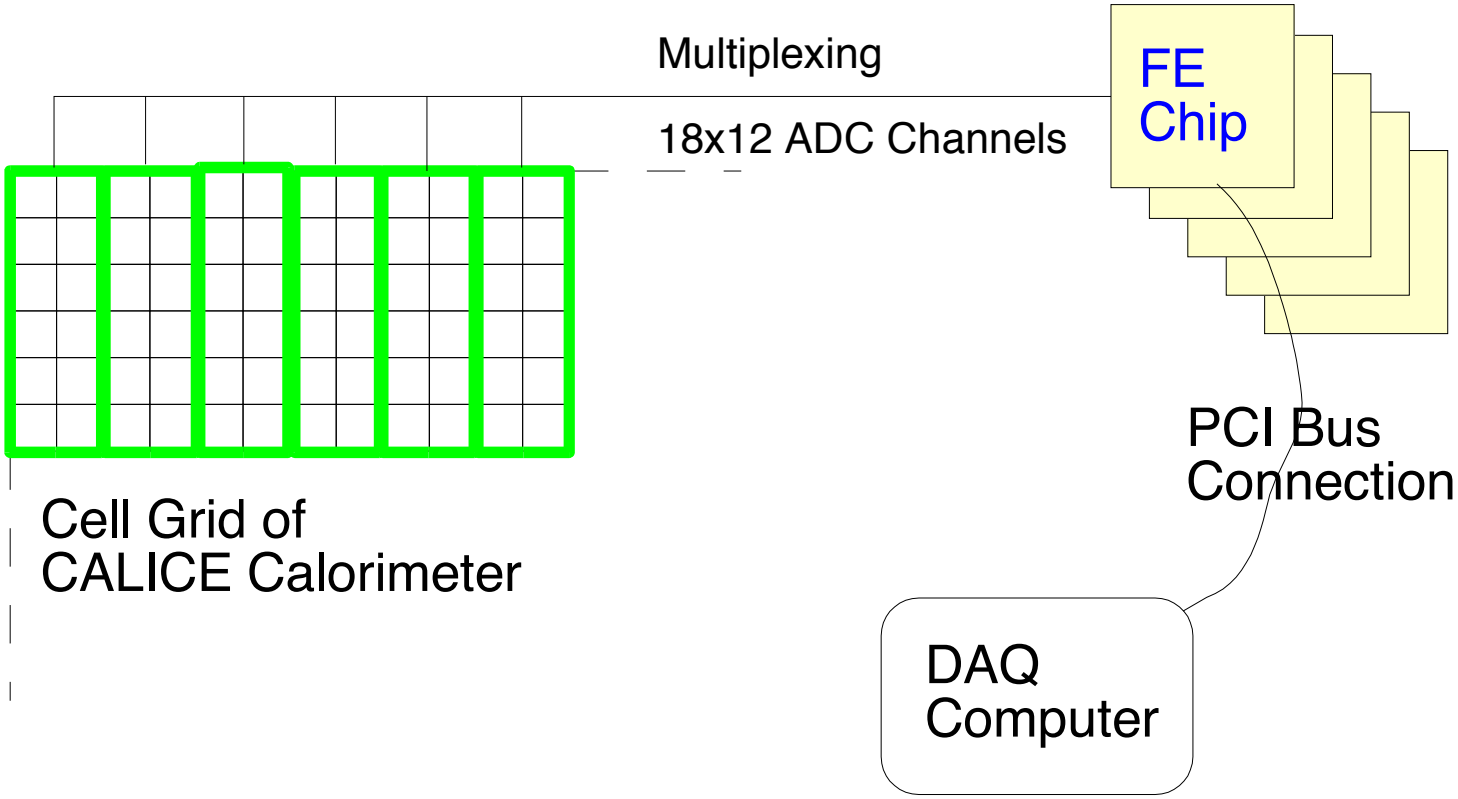
All analysis steps are to be realized within MARLIN frame

Dataflow in CALICE Testbeam Program

LCIO as backbone of Testbeam Analysis



CALICE DAQ Scheme – Poor Man's View



- DAQ is organized 'hardware friendly'
  Data received and stored as sequence of 32 bit integers
  Needs expert knowledge to analyze the data

- 'Many' people should get involved in calibration/monitoring

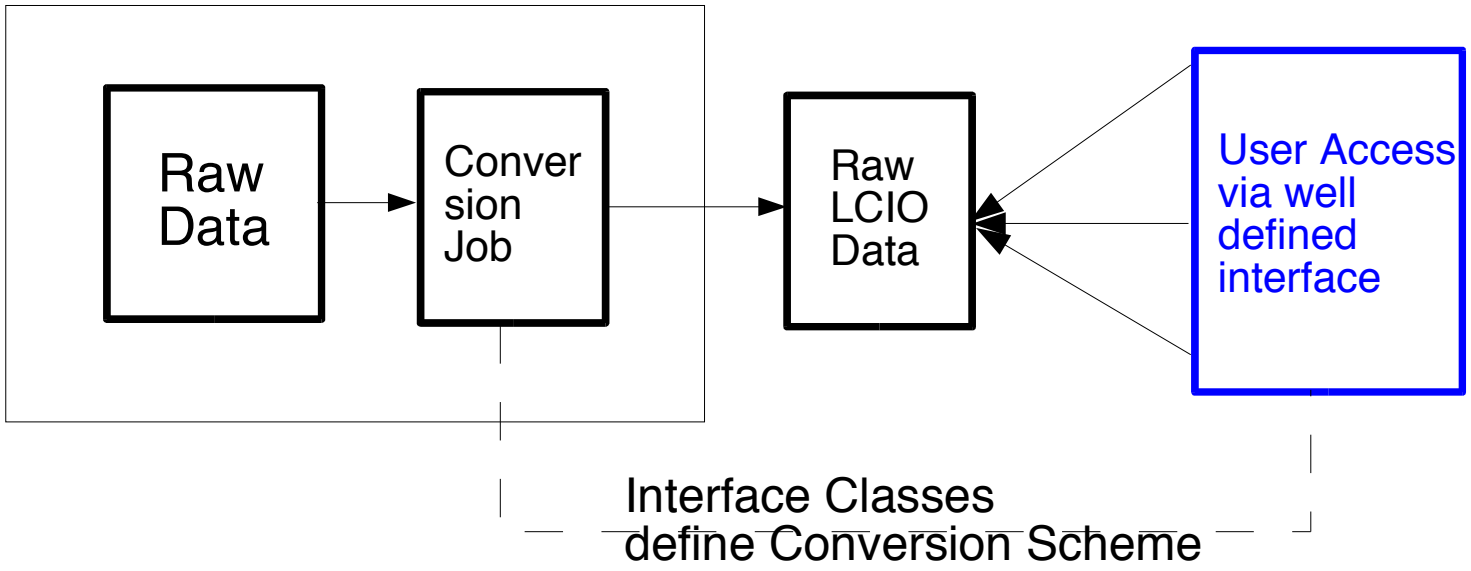
- Provide data in 'user friendly' format

Raw Data Conversion to LCIO

Collaboration: G. Mavromanolakis,D. Ward, P. Dauncey, F.Gaede, R.P.

Main Strategic Decision: Raw Data should be available in LCIO Format

Requirements: - 'Intelligent' Conversion from Raw Data to LCIO Raw Data
- Provide all Info on Raw Data also in LCIO Raw Data in a user friendly way

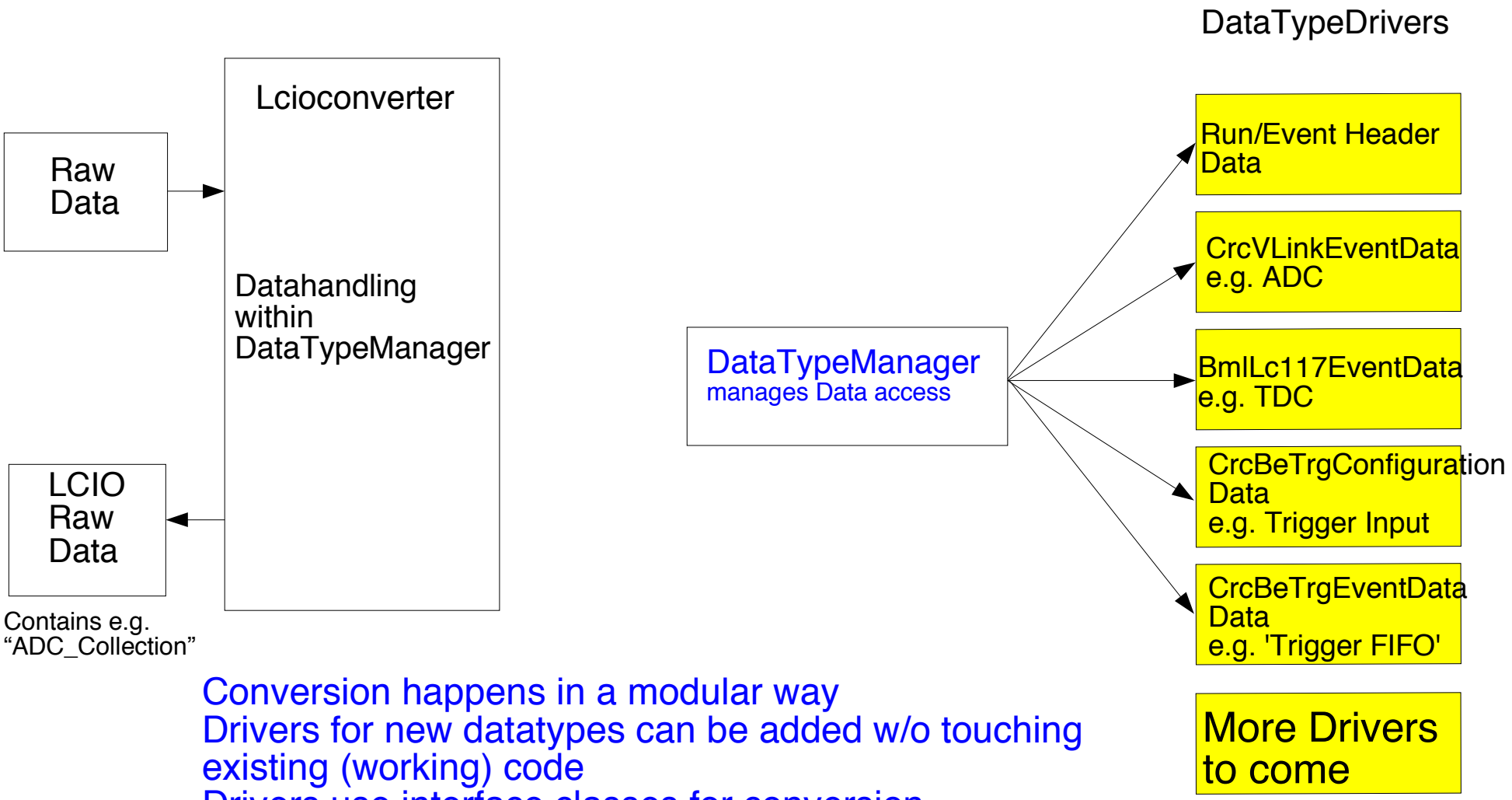


Interface is to be completely decoupled from online software

Dedicated Interface Classes are defined using LCIOGenericObjects

(Current) Structure of the LCIOConverter

Documentation: http://www-flc.desy.de/store/hcal/simsoft/calice_soft/lcioconverter/v00-05/doc/



Conversion happens in a modular way
Drivers for new datatypes can be added w/o touching existing (working) code
Drivers use interface classes for conversion

Handling of Conditions Data

Application of LCCD for Testbeam Data Processing

- LCCD – Linear Collider Conditions Data Framework:
  - Software package providing an Interface to conditions data
    - database
    - LCIO files

    Author Frank Gaede, DESY

- Conditions Data:
  - all data that is needed for analysis/reconstruction besides the actual event data
  - typically has lifetime (validity range) longer than one event
    - can change on various timescales, e.g. seconds to years
    - need for tagging mechanism, e.g. for calibration constants

ConditionsDBMySQL – Overview

Interfaced to LCCD by Frank Gaede

- Open source implementation of CondDB API
  - Conditions data interface for ATLAS

- developed by Lisbon Atlas group
- features
  - C++ interface to conditions database in MySQL
  - data organized in folder/foldersets
  - objects stored as BLOBs (binary large objects)
    e.g. LCIO objects or std::vector .....
  - tagging mechanism similar to CVS
  - scalability through partitioning options
  - outperforms implementation based on Oracle

ConditionsDBMySQL – Versioning of Conditions Data

Figure taken from CondMySQL Documentation

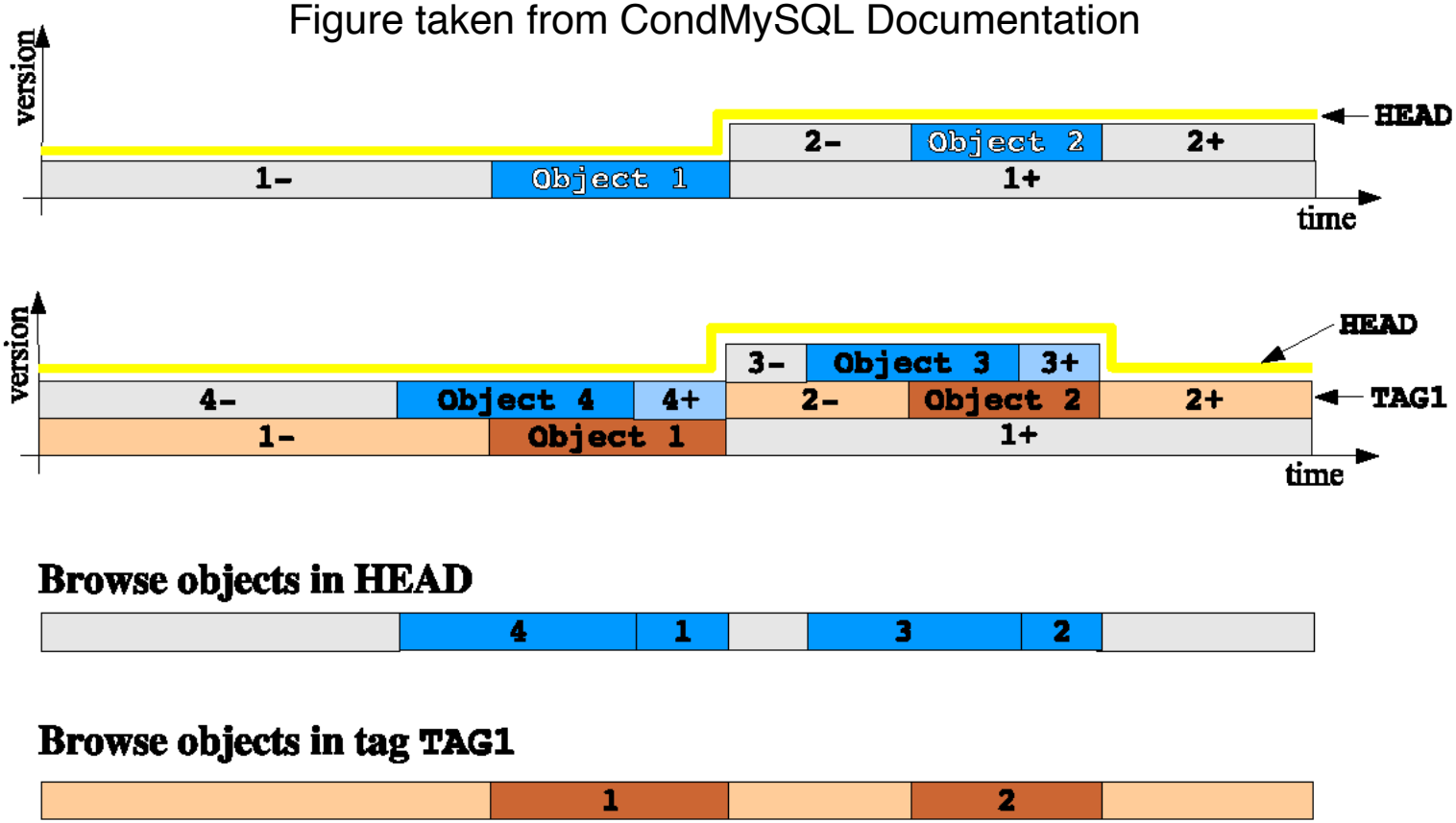


Figure 3: tagging and browsing example in the ConditionsDB mySQL's implementation.

CVS-like versioning/tagging system

'Horizontal' and vertical browsing in time possible

Time Stamp (by LCCD) in units of nanoseconds

ConditionsDBMySQL – Folder Organization

fld_id	fparent	insert_t	fpath	fdesc	fattr	ddtype	db_id	is_set
1	0	20050520000121	/			0	1	1
2	1	20050520000121	/lccd_calice			0	1	1
3	2	20050520000121	/lccd_calice/CellMapHcal			0	1	0
4	2	20050520000948	/lccd_calice/TriggerAssignment			0	1	0
5	2	20050520000956	/lccd_calice/TriggerCheck			0	1	0
6	2	20050520003030	/lccd_calice/BeTrgConfigData			0	1	0
7	2	20050520004534	/lccd_calice/BeTrgConfigLocation			0	1	0
8	2	20050520112113	/lccd_calice/CellMap			0	1	0

...

- UNIX-Like Tree Structure
- Each Folder Contains one set of ConditionsData
  CellMap relation Electronic Channel ↔ CellID
  Stored as set of LCFixedObjects
  LCCD provides Streamer Methods to read LCIO data types back from DB
- Access via Folder Name

Acessing ConditionsData Using LCCD – Users Point of View

MARLIN versions > 00-07 prepared to deal with Conditions Data
(Note: LCCD does not depend on MARLIN and vice versa)

- Source of ConditionsData defined in MARLIN steering File
  e.g. ConditionsData for Cell Mapping from DB

```
DBCondHandler channelmap /lccd_calice/CellMap V00-01
```

- Handling of Conditions Data (updating etc.) within a ConditionsProcessor (provided by MARLIN)

- Code is completely transparent to Conditions Data Source
  a) Register Pointer to a CellMap and its name

```
lccd::LCConditionsMgr::instance()->registerChangeListener( _cellMapPtr , “channelmap” );
```

  b) Obtain CellMap within event

```
int cellID = _cellMapPtr->find( elec_channel ).getCellID() ;
```

  c) That's all ...

Identified Conditions Data for CALICE

- Calibration Constants
- Cell Mapping Schemes
- Meta information about SiPMs (HCAL)
- Trigger Configuration and validity
  Trigger Information changes even during runs

- N.B.: TriggerHandlerClass for CALICE lives as a singleton Class within MARLIN (not designed as own processor)

Data Storage Issues

4 weeks of CALICE Ecal data taking ~ 250 Gbyte (unconverted) data
All data have are stored in the DESY dCache pool !

- Infrastructure for mass storage setup at DESY (dCache Pool)
  Roughly 5 Pbyte diskspace available
- Recommended access via Grid-Tools
  Not very much experience gathered yet
  Mastering this is essential for sharing of CALICE and ILC data
  CALICE VO hosted by DESY
- dccp tool installed on DAQ Computer in Control Room at DESY
  ⇒ Immediate copy of data into dCache possible
- dCache is common effort of DESY and Fermilab !
  ⇒ Similar 'storage' environment at Fermilab

See also: `http://www.dcache.org/`

Recent Developments

Most important data information can be converted to LCIO

- Run, Event and Timing information
- Trigger information
- ADC Data

Data of AHCAL are regularly converted into LCIO and stored on the dCache pool

- Development of dedicated analysis algorithms under way

MARLIN based reconstruction exists for Ecal

- still in testing phase
- Conversion within MARLIN ?
- Online histos within MARLIN ?
- non MARLIN based tool working on converted files exists, too

Efforts to establish a common database at DESY

- Will exploit new FLC webserver for that task

Pieces of Software will be put together by end of July beg. August

- Expect working chain for Vienna workshop
- sorry not for Snowmass (Too heroic effort)

The CALICE cvs repository

Set up by H. Vogt DESY Zeuthen
Find it here: `www-zeuthen.desy.de/linear_collider`
No account at Zeuthen needed, safe access via ccvssh

Snapshot of Webinterface

File	
CVSROOT/	
calice/	Pool for misc. software goodies and tools
calice_lcioconv/	Contains lcioconverter
calice_online/	
calice_reco/	Contains userlib (i.e. Interface classes)
calice_sim/	Contains Hcal Ganger
lcsoft/	
log/	
test_project/	

Conclusion and Outlook

- CALICE Testbeam software follows closely and puts demands on software development for ILC related studies
- Maybe the first project which makes full use of existing ILC software tools (Mokka, LCCD, MARLIN ...)
- Conversion of Raw Data to LCIO Format
- LCCD is powerful tool to handle Conditions Data
  Storage of Conditions Data in MySQL Database !?
  LCCD allows also for other sources
- First (small) software chain to process testbeam data using LCIO/MARLIN/LCCD exists
- Not discussed but, need to look at the digitisation step soon
  Exploit DigiSim Package of G. Lima !?

Next Step: Assemble pieces into working machinery

Final Remarks

Analysis of CALICE data depends on five++ software packages
All tools are really useful !!!!!!

- LCIO
- MARLIN (heavily advancing)
- LCCD
- CondDBMySQL (+MySQL)
- Mokka (+GEANT4)
- Maybe GEAR in the near future

In addition we have JAS, WIRED, Root, AIDA ....

This sometimes frightens users
Not so lightweight after all

Maybe there is no way out and users have to swallow this

What about setup tools for ILC software ?
I know that this is difficult but don't we need it ?