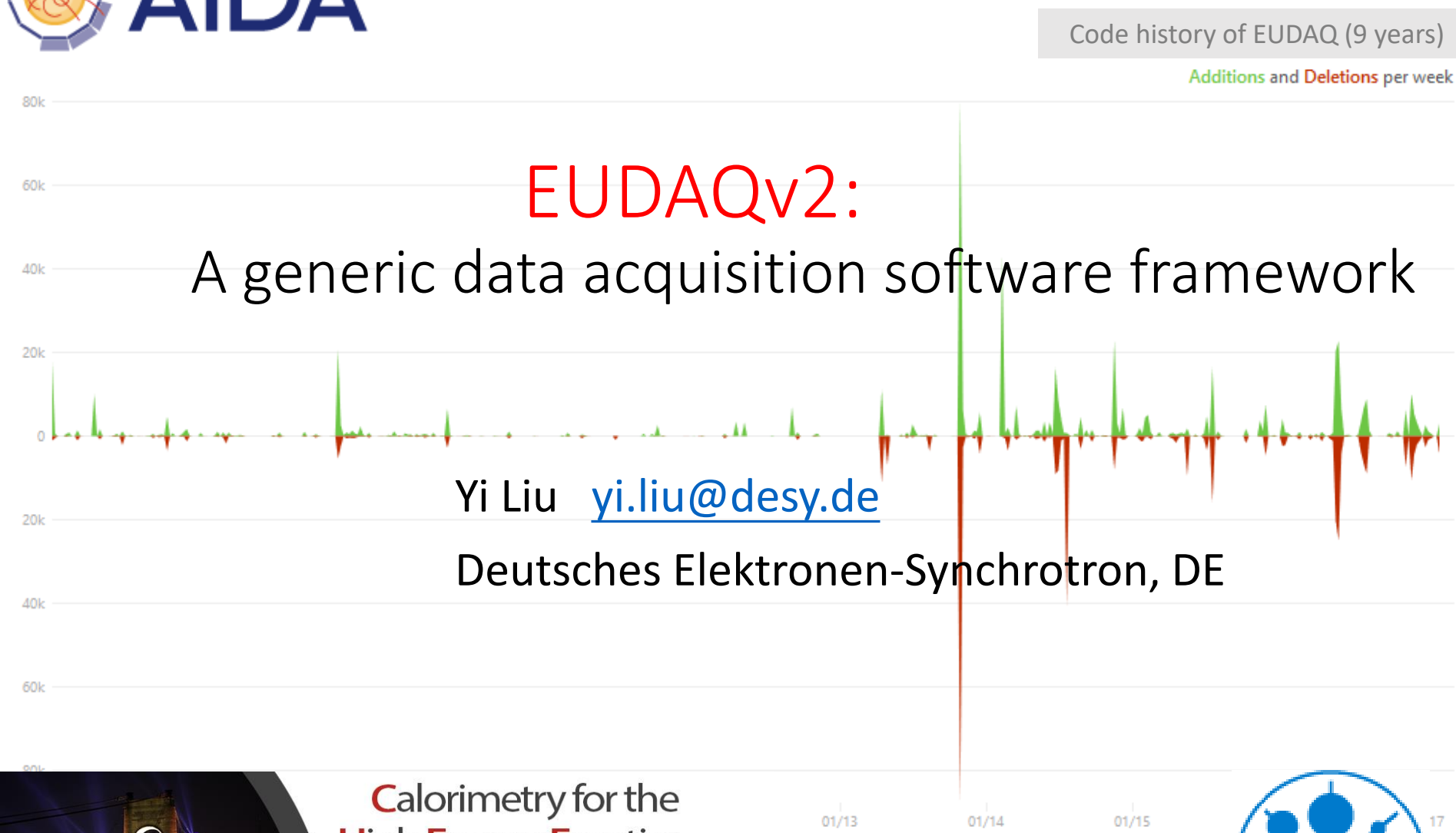
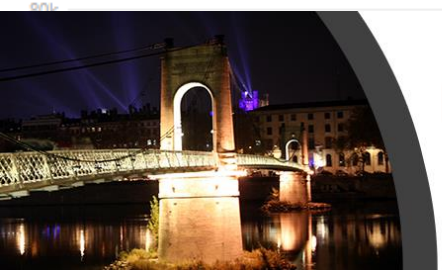


EUDAQv2:

A generic data acquisition software framework

Yi Liu yi.liu@desy.de

Deutsches Elektronen-Synchrotron, DE



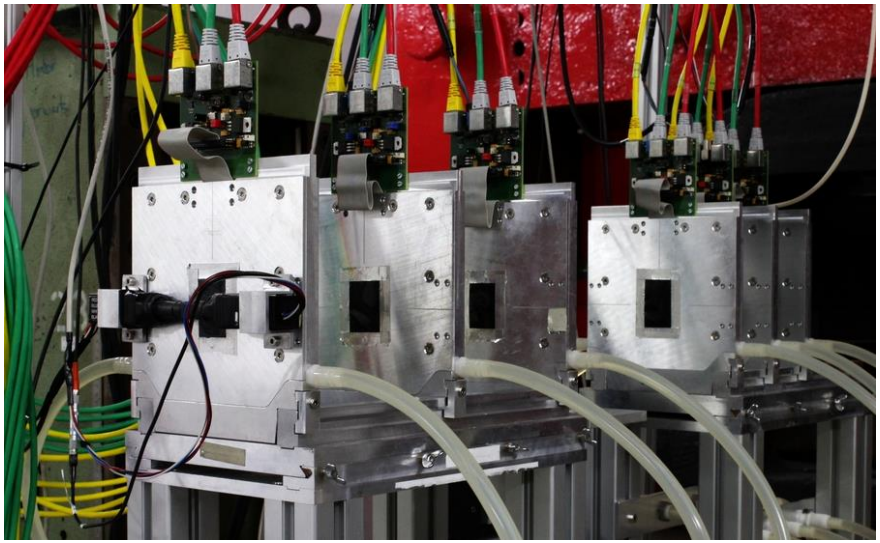
Calorimetry for the High Energy Frontier

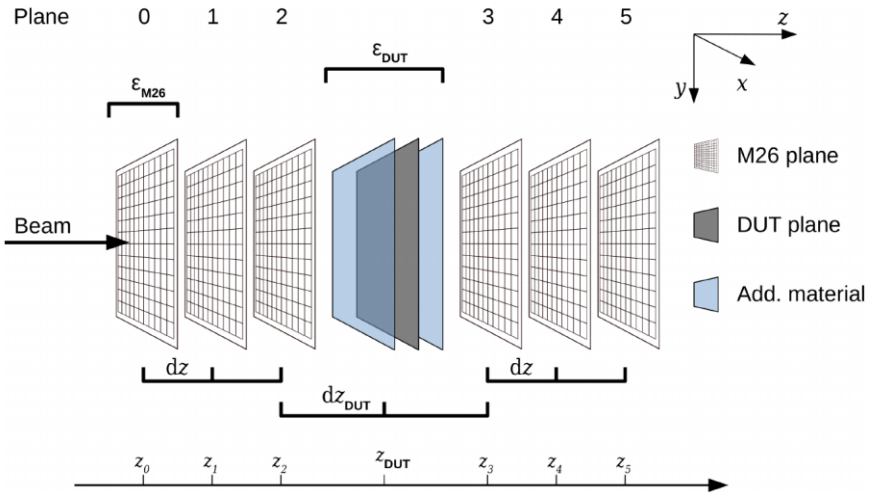
Lyon, France

2 - 6 October 2017

History: EUDAQv1 on EUDET telescope

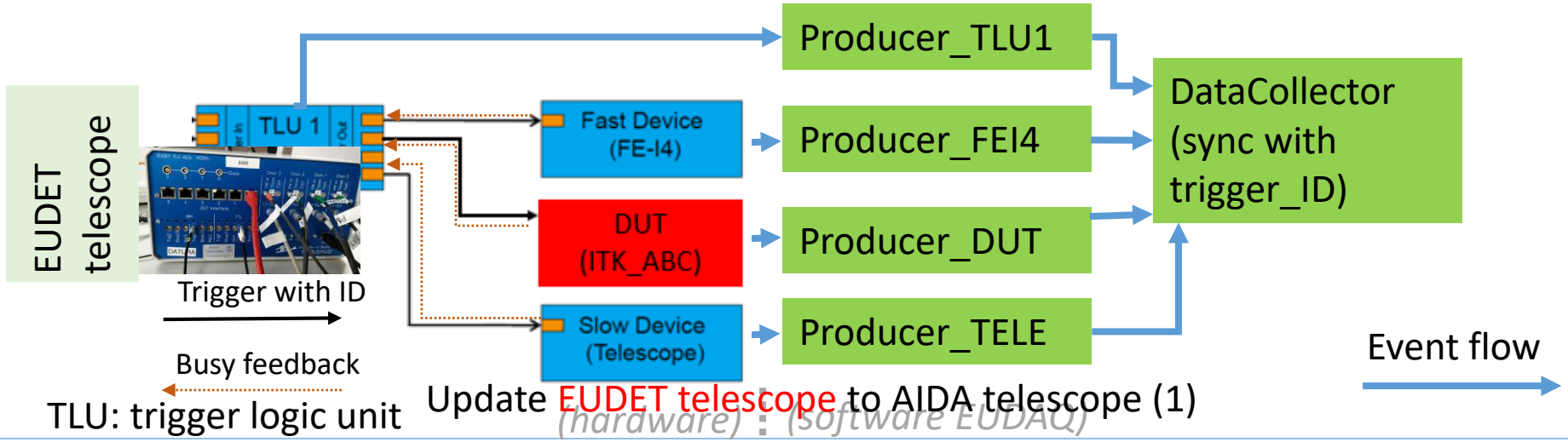
- EUDET telescopes are used heavily at the test beam facilities for the detector prototypes. (EUDET copies in CERN, DESY, Bonn)
- EUDAQ is originally developed as a DAQ system for EUDET-type telescopes.
- EUDAQ provides centralized controlling, logging.
- EUDAQ provides an interface for 3rd part users who do beam test for the their detector prototype under EUDET telescopes.





EUDET telescope https://telescopes.desy.de
18.5um pixel pitch
3um tracking resolution by 6 Mimosa26 planes

Motivation:
Update EUDET telescope to AIDA telescope



AIDA

EUDET telescope

- A system trigger signal with trigger-ID is distributed in all telescope sub detectors.
- Sub detector reads trigger-ID and insert it to a triggered sub event.
- Trigger-ID is the key to merge sub events.

Motivation:
Update EUDET telescope to AIDA telescope

EUDET

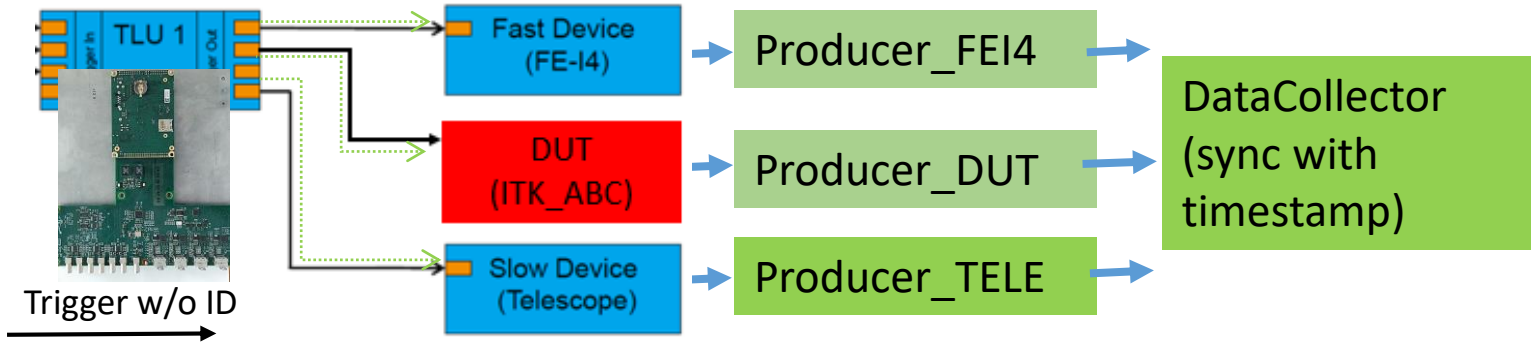
AIDA telescope

- A system clock will distribute to all telescope sub detectors.
- Sub detector counts the clock circle to generate timestamp and insert it to a triggered sub event.
- Timestamp is the key to merge sub events.

TLU: trigger logic unit

(hardware) ⋮ *(software EUDAQ)*

AIDA telescope



All hardware needs to be updated to accept system clock instead of Trigger ID.

Extend its use case as common DAQ

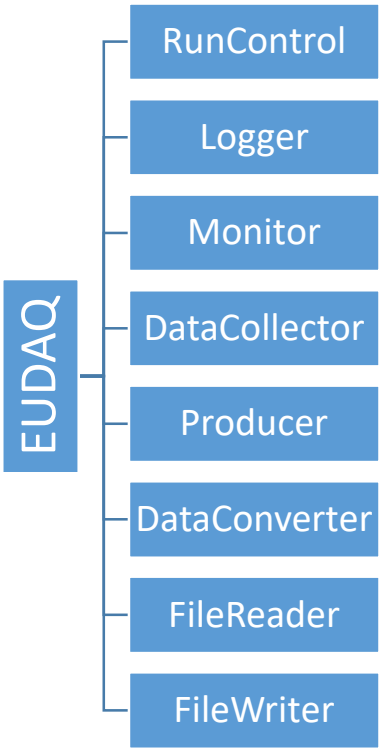
- Key features to be a common DAQ
  - Distributed data taking
  - Central Control and configure interface.
  - Data collector/builder and data converter
  - GUI, Monitor
  - Extensible
  - Cross platform

In EUDAQv1
√
√
√ *
√ *
√
√

- EUDAQ1 has almost all required key features to be a common DAQ, (*) except its data collector and Monitor was designed for EUDET hardware
- EUDAQ2 is a major version release. Let's take this chance to make a significate change of interface and improve to a nicer/modern code.

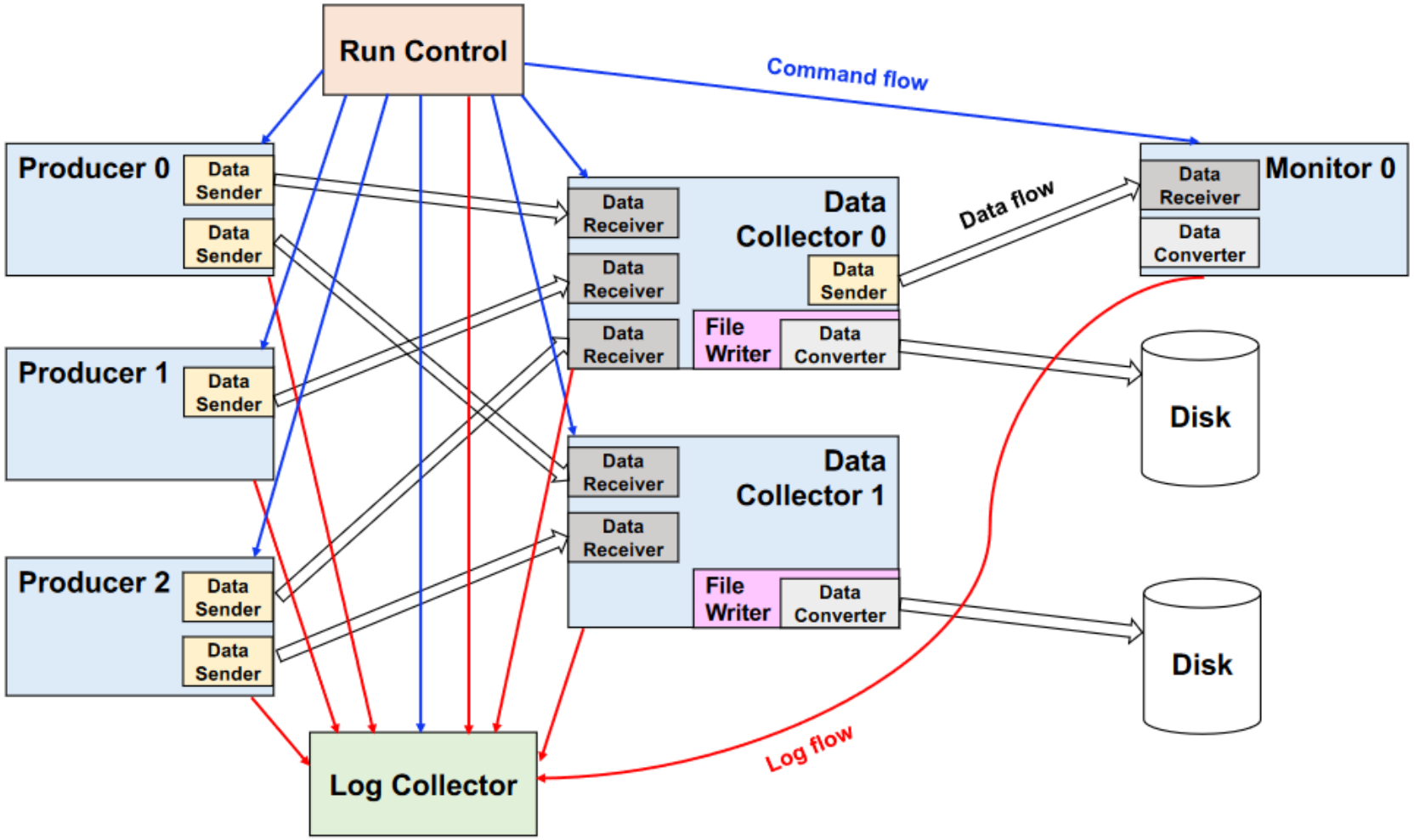
Extend its use case as common DAQ

EUDAQ2 consists of several components, which run online for data-taking or offline for data converting and quality analysis.



	EUDAQ2 Executable	Run mode
Control all EUDAQ components	euRun, euCliRun	online
Log the message	euLog, euCliLog	online
Display hit information online	OnlineMon(EUDAQv1)	online/offline
Merge the sub events.	euCliDataCollecctor	online
Talk to device and send sub events	euCliProducer	online
Convert data	euCliDataConverter	online/offline
Read events from a stored file	euCliReader	offline
Format and write events to file	none	Internally

Command / Data flows of EUDAQ2 on network



Schematic of the EUDAQ2 architecture (C++ encapsulation and network talking)

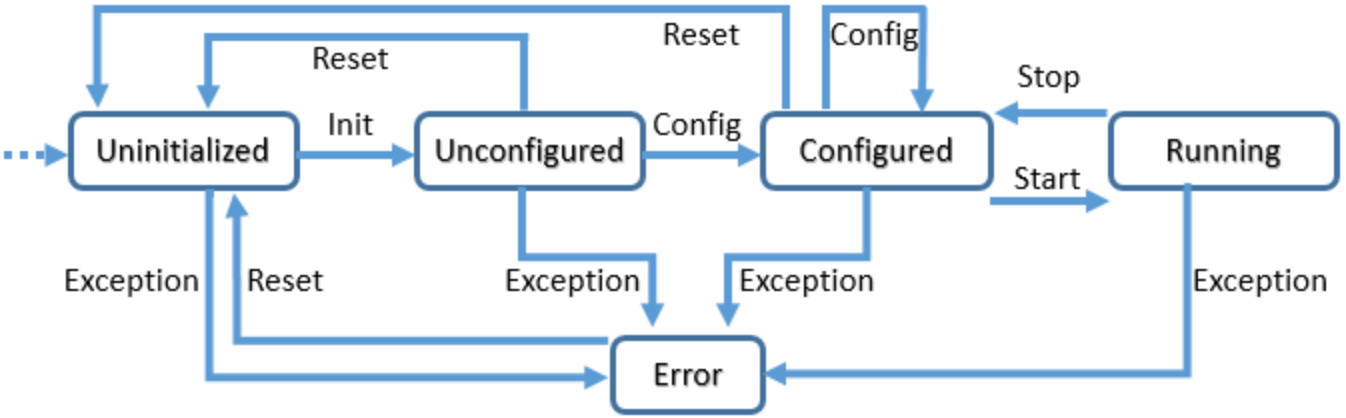
Component: RunControl

RunControl maintains a database about the address of clients and sends command to them.

- The Standard RunControl EUDAQ is enough for most user with a simple setup.

New in EUDAQ2

- QT GUI is decoupled from EUDET-Telescope RunControl
- User can reuse the GUI with their own RunControl without touch GUI code.
- Provide flexible to have dedicated RunControl to integrate with other DAQ system .
- The FSM states EUDAQ clients are checked by RunControl



FSM state

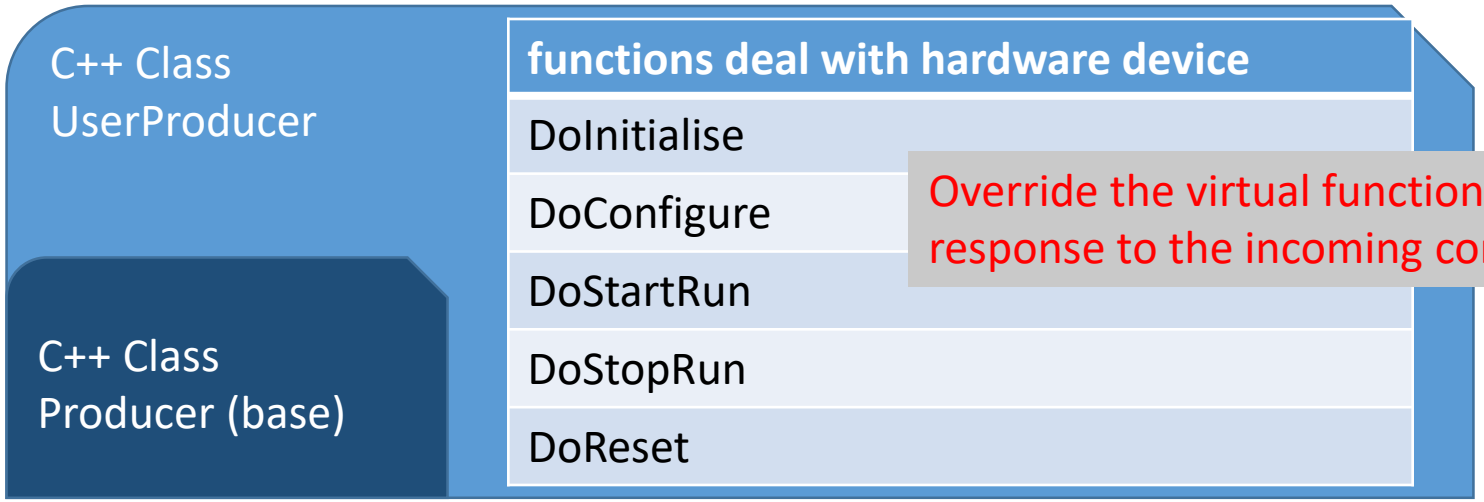
Component: Producer

Producers are the binding part between a user DAQ and the central EUDAQ RunControl.

- The base Producer do all the common tasks for the derived Producer to simplify the integration.

New in EUDAQ2

- Unique launch executable application (euCliProducer)
- Runtime name
- FSM state is managed internally
- Configurable Data sending destination.



Component: DataCollector

The Data Collector receives all the data streams from all the Producers, and combines them into a single stream.

- Capable to event data to different file formats by configuration.

New in EUDAQ2

- Unique launch executable application (euCliDataCollector) and runtime naming.
- FSM state is managed internally.
- Event type independently.
- Multiple DataCollectors at one running setup

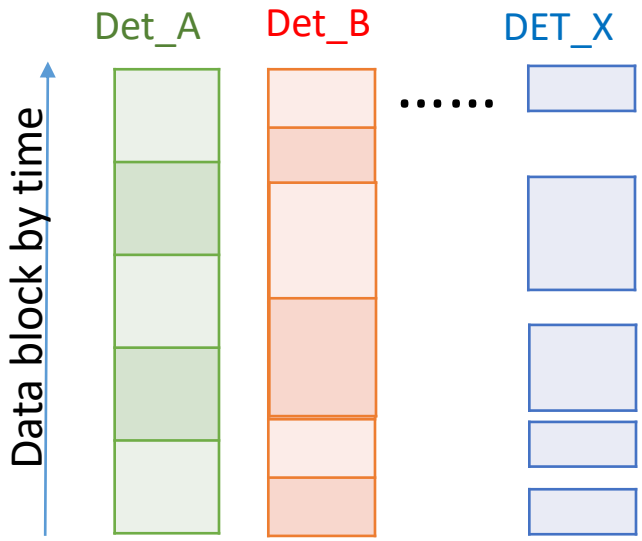
C++ Class UserDataCollector

C++ Class DataCollector (base)

Cmd Functions	Data Functions
DoInitialise	DoConnect
DoConfigure	DoDisconnect
DoStartRun	DoReceive
DoStopRun	
DoReset	

Component: DataCollector (DoReceive)

Only the detector developers (EUDAQ users) know the timing relationship among the sub-detectors.



In generic case, sub event(data) can be generated, by trigger, by roll-shutter-readout, at any anytime.

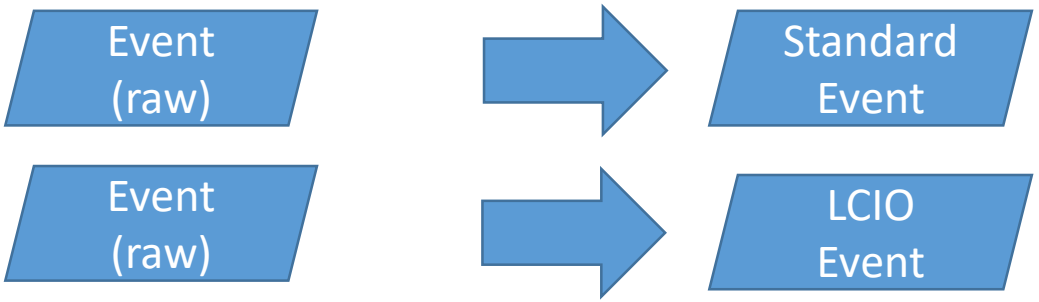
How to defined a physical event?
How to defined a valid trigger?
Which detector is the trigger device?

Assuming each data block have has a timestamp pair for the begin and end of trigger/readout.
fix/variable time slice (Det_A/Det_B)
continues/discontinuous (Det_X)

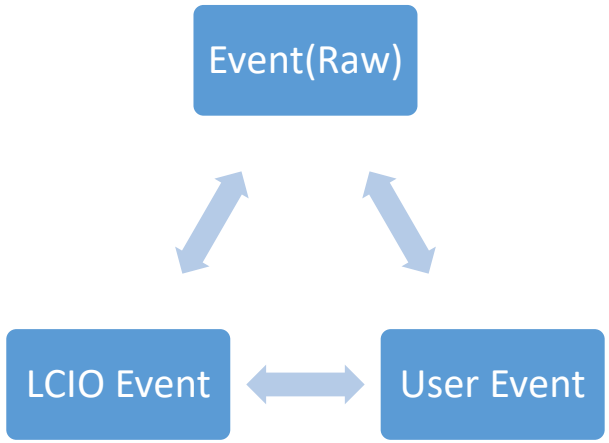
- Example DataCollectors are provided: sync by timestamp/trigger number.
- It is open to user to implement a DataCollector for a specific hardware setup.

Component: DataConverter

- Modular designed for conversion between any kinds of data types.
- Conversion from EUDAQ Event to LCIO Event is native supported.
- DataConverter can be called online by DataCollector/Monitor if they want data in another format instead of EUDAQ native format.



User can extend the supported Event type of EUDAQ framework. (optional bidirectional converting)



Possible circle converting, if the correlated DataConveters among those event types are all implemented

Modular building / Runtime discovery

EUDAQ2 are build to 3 categories of libraries

Binary	Category	Source Code	Description
libeudaq_core	Core	main\lib\core	core library
libeudaq_lcio	Extension	main\lib\lcio	lcio extension library
libeudaq_std	Extension	main\lib\std	standard extension library
libeudaq_module_test	Module	main\module\test	test module library
libeudaq_module_example	Module	user\example\module	module library by example user

1. Core: the core library. It should be always build and installed.
2. Extension: optional features of EUDAQ (eg. support external data format).
3. Module: Dynamically discovered and loaded by EUDAQ core library at run-time.

The hardcode path dependence is removed.
We are going to distribute EUDAQ as binary package to End-User.
(by RPM, DEB, Windows installer)

Python and ROOT interface (work in progress)

Python:

Pybind11 provides a solution to export C++11 symbols to python.

Analysis offline raw data

Implement data-taking program by python

```
>>> import libpyeudaq as eudaq
>>> ev = eudaq.Event("demo_raw")
>>> ev.SetEventN(1)
>>> ev.SetTriggerN(2)
>>> ev.SetTimestamp(1600,1610)
```

In PYTHON Interpreter
import EUDAQ library
and then create an Event

print the event

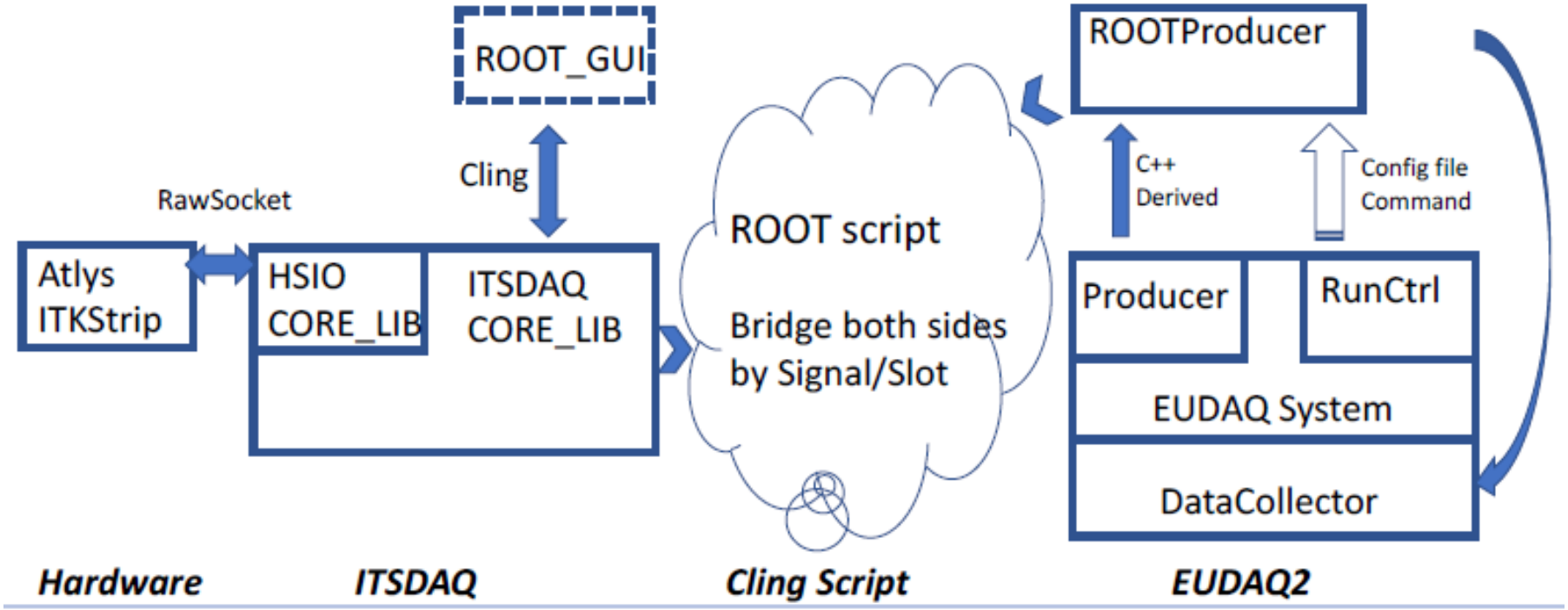
```
>>> print ev
<Event>
  <Type>2149999981</Type>
  <Extendword>3068292765</Extendword>
  <Description>demo_raw</Description>
  <Flag>0x00000030</Flag>
  <RunN>0</RunN>
  <StreamN>0</StreamN>
  <EventN>1</EventN>
  <TriggerN>2</TriggerN>
  <Timestamp>0x0000000000000640  ->  0x0000000000
  <Timestamp>1600  ->  1610</Timestamp>
  <Block_Size>0</Block_Size>
</Event>
```

ROOT:

ROOT dictionary can be generated and shared library can be load into ROOT.

Useful feature to analysis data and plot graph with EUDAQ available in ROOT interpreter (cling)

User Case: ATLAS ITK strip beam test



ITSDAQ and EUDAQ export their interface to ROOT/Cling.
Their dynamic shared libraries are then loaded in ROOT scripts.

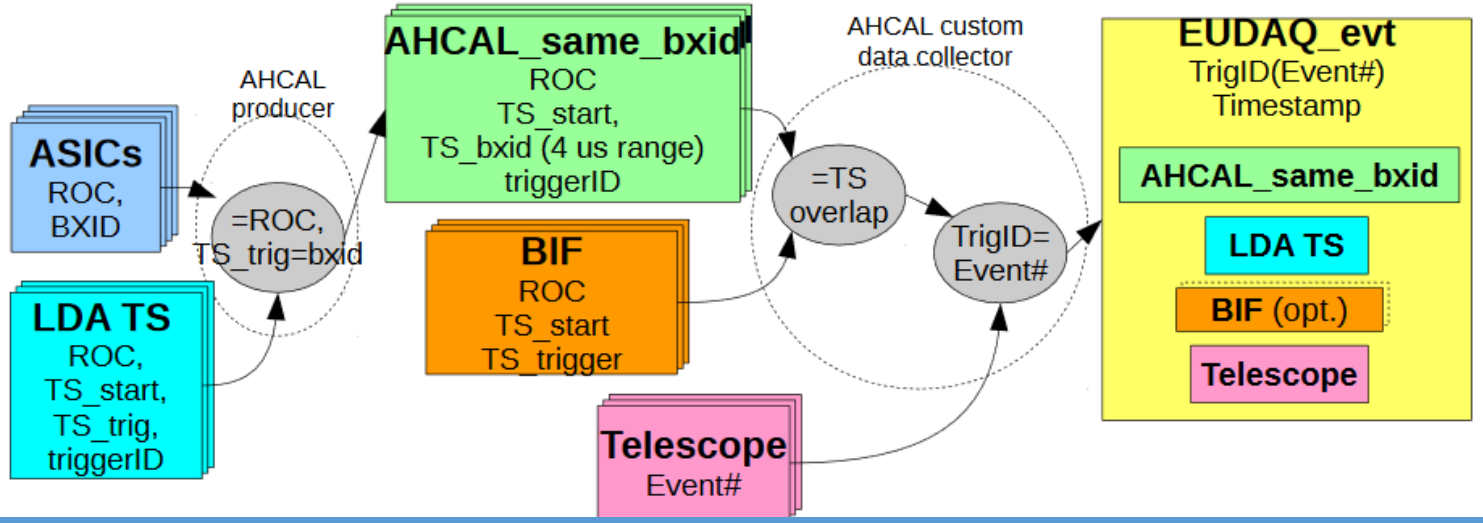
Signal/Slot mechanism is adopted to crossover both sides.

User Case: CALICE AHCAL beam test

CALICE AHCAL as the first user of EUDAQv2 verified the designed EUDAQv2 functionality.

- AHCAL: ROC + BXID in ROC
- LDA: ROC + TS_ROC_Start + TS_trigger + TriggerID
- BIF: TS_ROC_Start + TS_trigger
- Telescope: TriggerID
- TLU: TriggerID

The timestamps come from separated clocks.
The user DataCollector implements its method to detect fake event, align the clock offset.

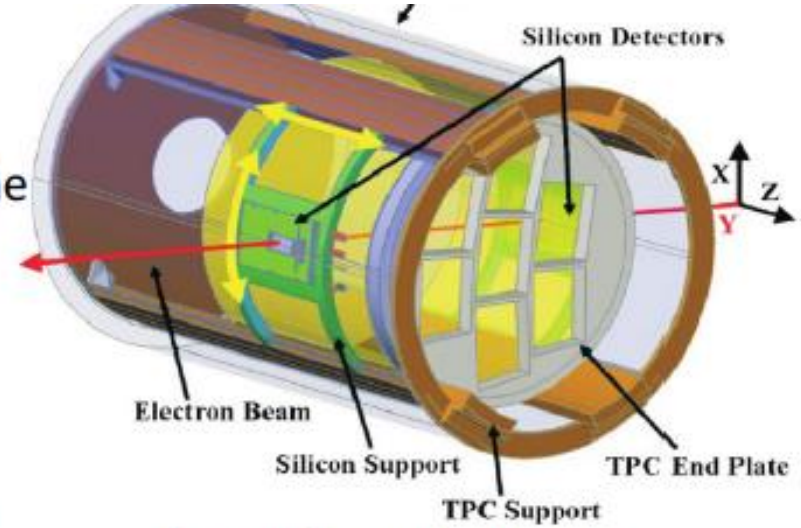


User Case: DESY Strip Telescope

DESY test beam infrastructure T24

Installed in magnet for TPC beam tests with the ECal sensor bump-bonded to KPiX
Large area sensors (~10x10 cm2) with an expected spatial resolution of 7-8µm

KPix itself has a readout DAQ which runs as a **slaver system under EUDAQ2**.



The mechanical support (PCMAG 1T solenoid)

Code Maintaining

- EUDAQ is applied by GNU Lesser General Public License

- Codes is available on GitHub repository

https://github.com/eudaq/eudaq

- Continuous integration to check compile time error of the code.
  Linux/MacOS hosted on Travis CI
  Windows hosted on AppVeyor

- Preparing the binary distribution to end users.

- User Manual is available:

http://eudaq.github.io/manual/EUDAQUserManual_v2_0_1.pdf

Summary

- EUDAQ is developed for EUDET telescope.
- The EUDAQv2 has a significate change of API interface.
  - Simple, Explicit, Extensible
- EUDAQv2 Core library is independent from EUDET telescope.
- Cross Platform of OS and binary distribution
- Python and ROOT interface (WIP)
- Early users:
  - EUDET telescope and its upgrade
  - CALICE/AHCAL test beam
  - ATLAS ITK strip test beam
  - strip telescope at TB24 (WIP)
- Both the user manual and example code are available

Thank you for your attention.