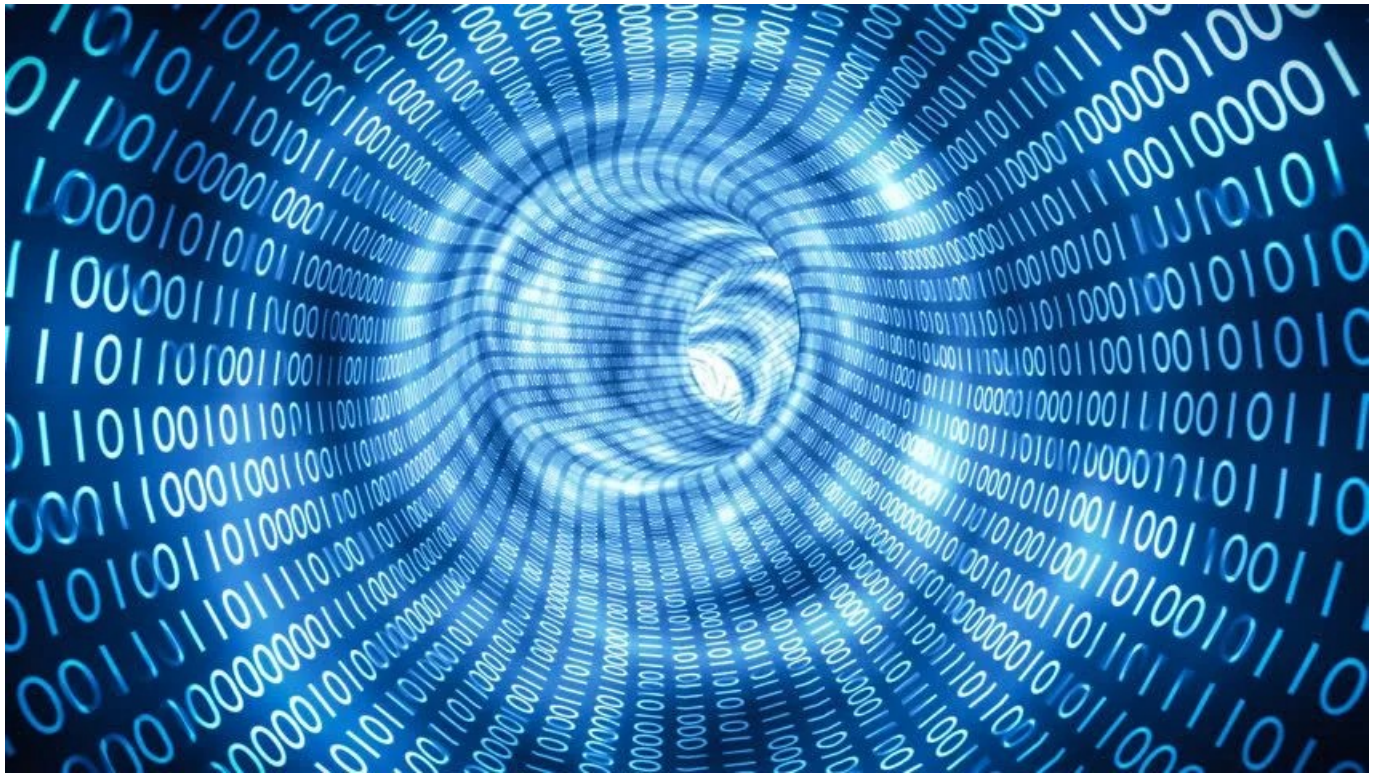
Introduction to Trigger and Data AcQuisition

3/3 – full TDAQ control



Plan

- **Basic DAQ concepts**
-
- **TDAQ Infrastructure AND FOOT use**
-
- **TDAQ Advanced**

What we will see today

- Remote detectors

- Information service
- Online Histograms and Presenter
- GNAM
- MYSQL tables

- FOOT components

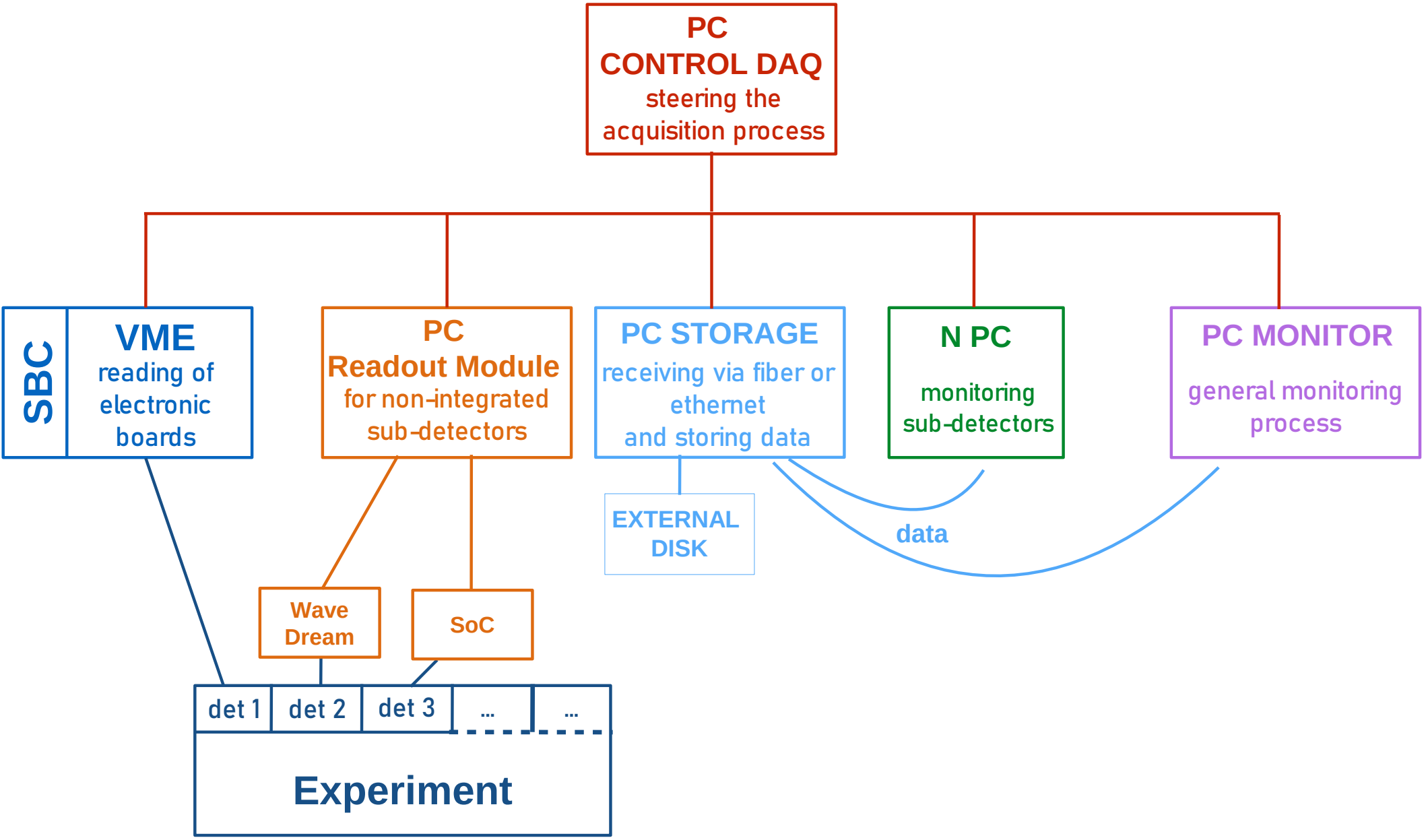
“Remote detectors”

In FOOT language a remote detector is a detector whose front-end is not read out **directly** by a TDAQ readout Module.

The global infrastructure is NOT present on the remote detector…. No need to be a linux machine….

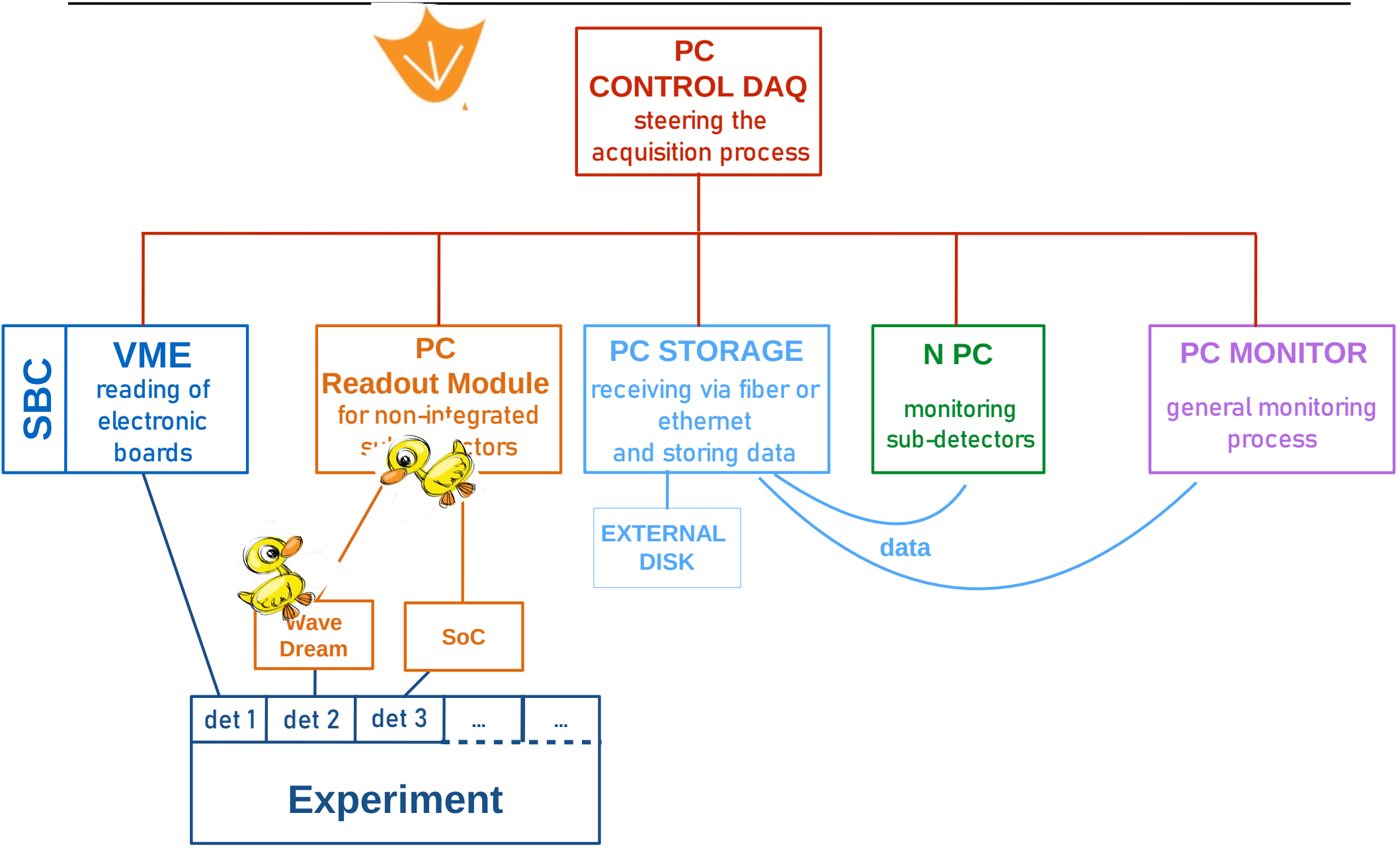
We need to mimick the basic TDAQ functionalities on the “remote machine”

System overview



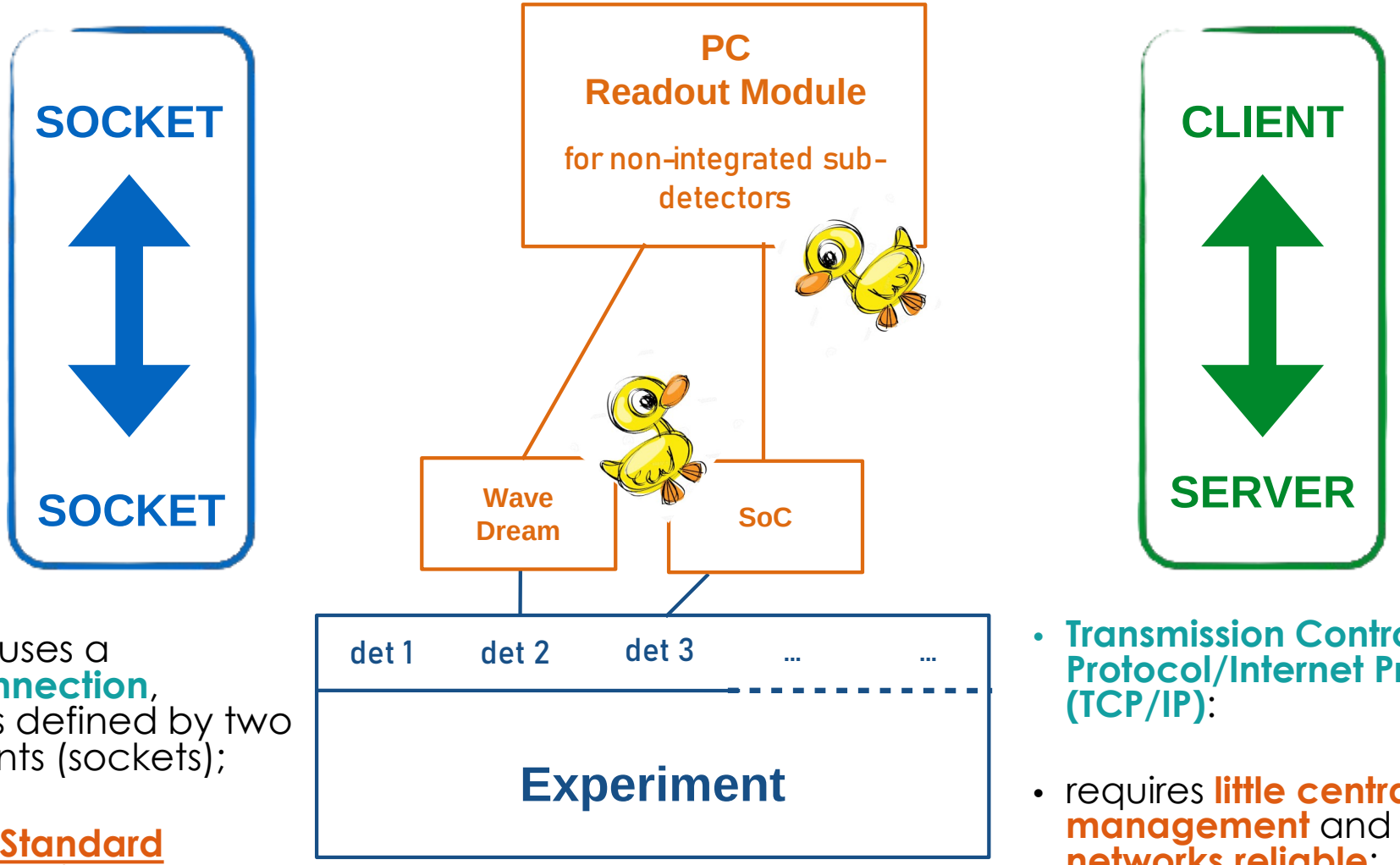
Credits: Silvia

System overview



Credits: Silvia

Connection via TCP/IP



- Socket uses a **TCP connection**, which is defined by two endpoints (sockets);
- all **C++ Standard Template Library based**;
- provide **reliable two-way communication**.

- **Transmission Control Protocol/Internet Protocol (TCP/IP)**:
- requires **little central management** and makes **networks reliable**;
- IP is **compatible with all operating systems and with all types of computer hardware and networks**.

Credits: Silvia

Basic scheme

GLOBAL TDAQ

Readout module = CLIENT

1. send **configuration** parameters to server
2. request **monitoring parameters** from server
3. tell the server **when to send data**

connections going from the same client to the same server

"Slow" connection

starts when "configure" mode and uses > 1 fork to handle different simultaneous activities

DETECTOR SIDE

Sub-detector = SERVER

1. receive **configuration** from client
2. send **monitoring parameter** to client
3. set a device parameter to "run" status and **prepare itself for the data flux**

Credits: Silvia

GLOBAL TDAQ SIDE (client)

- A `ModuleRemote` object having an instance of a `DataChannelRemote` object is all that is needed.

Detector SIDE (data server)

A “Readout Application” clone is present:

```
class TDAQServerBase {
public:
  TDAQServerBase(std::vector<std::string> & arguments, DAQServerInterface* servInt);
  virtual ~TDAQServerBase();
  virtual void runServer();        // actual call to run the serve
   virtual void configure();          // TDAQ transitions
  virtual void publish();
  virtual void run();
  virtual void stopDC();
private:
    DAQServerInterface *p_server;       // detector interface
    EventCircularBuffer* m_cb;          // where events are stored
};
```

Detector SIDE (detector interface)

```
class DAQServerInterface {
public:
  virtual int initialize();
  virtual int shutdown();
  virtual void configure(std::vector<uint32_t> & param);
  virtual void GoToRun();
  virtual void StopDC();
  virtual void publish(std::vector<uint32_t> & param,
                       std::vector<uint32_t> & results);

  virtual uint32_t wordsAvailable();
  virtual void readData(std::vector<uint32_t> & evt,
                        int maxValues);

protected:
  bool m_isRunning;
}

class DE10ServerInterface : public DAQServerInterface {
 ….
}
```

Controllable

Data Channel

Actual class

Remote modules

In FOOT we have remote modules for:

Wave Dream
(SC+TW+CALO+neutrons)

Vertex

Inner Tracker

MicroStrip Detector

Information Service

CORBA Service where to publish or to read information on the run, boards, system.

Data are stored locally as **objects** in **named** directories inside IS servers

Example: Server "RunParams" contains 4 object(s):
RunParams.RunInfo
RunParams.RunParams
RunParams.SOR_RunParams
RunParams.LumiBlock

Value	Type	Name	Description
3338	U32	run_number	ATLAS run number
0	U32	max_events	Maximum number of events
0	U32	recording_enabled	Recording enabled
0	U32	trigger_type	Trigger type for this run
V2495	String	run_type	Run type for this run
00000000000000000000000000000000	String	det_mask	String-encoded detector mask for this run
0	U32	beam_type	Beam type
0	S32	beam_energy	Beam Energy
tyrtyryd	String	filename_tag	Filename Tag for writing information to persistent storage
data_test	String	T0_project_tag	this fields holds the project tag that will be used at Tier 0 to seed the recons

data_test

Readout Module & publish()

Each readout module has a method publish() running in a separate thread that is called periodically (eg every 10 s). Slow control data can be read out and published in the Information Service
(do you remember data acquisition with a *periodic* read out?)

The IS server used for these data is «Monitoring»
The named directory where to publish data is usually related to the actual readout module name (eg. «TDCBoard1», «MSD_L2»)
The data to publish are stored in a (CORBA) object

Object definition

A schema is used to define the data to publish.

Foot example: file ModuleRemote.schema.xml

```
<class name="ModuleDE0Info" description="DE0nano1 parameters to show on IS">
 <superclass name="Info"/>

 <attribute name="BoardName" description="BoardName" type="string"/>
 <attribute name="PortNoSlow" description="Remote port number"  type="u32"/>
 <attribute name="HostName" description="Host name" type="string"/>
 <attribute name="FWversion" description=«Firmware Version" type="u32"
                format="hex" init-value="0x00000100"/>
```

<

C++ include file

With a suitable makefile the schema file is converted to a C++ include file:
Foot example: file ModuleDE10RemoteInfoNamed.hh

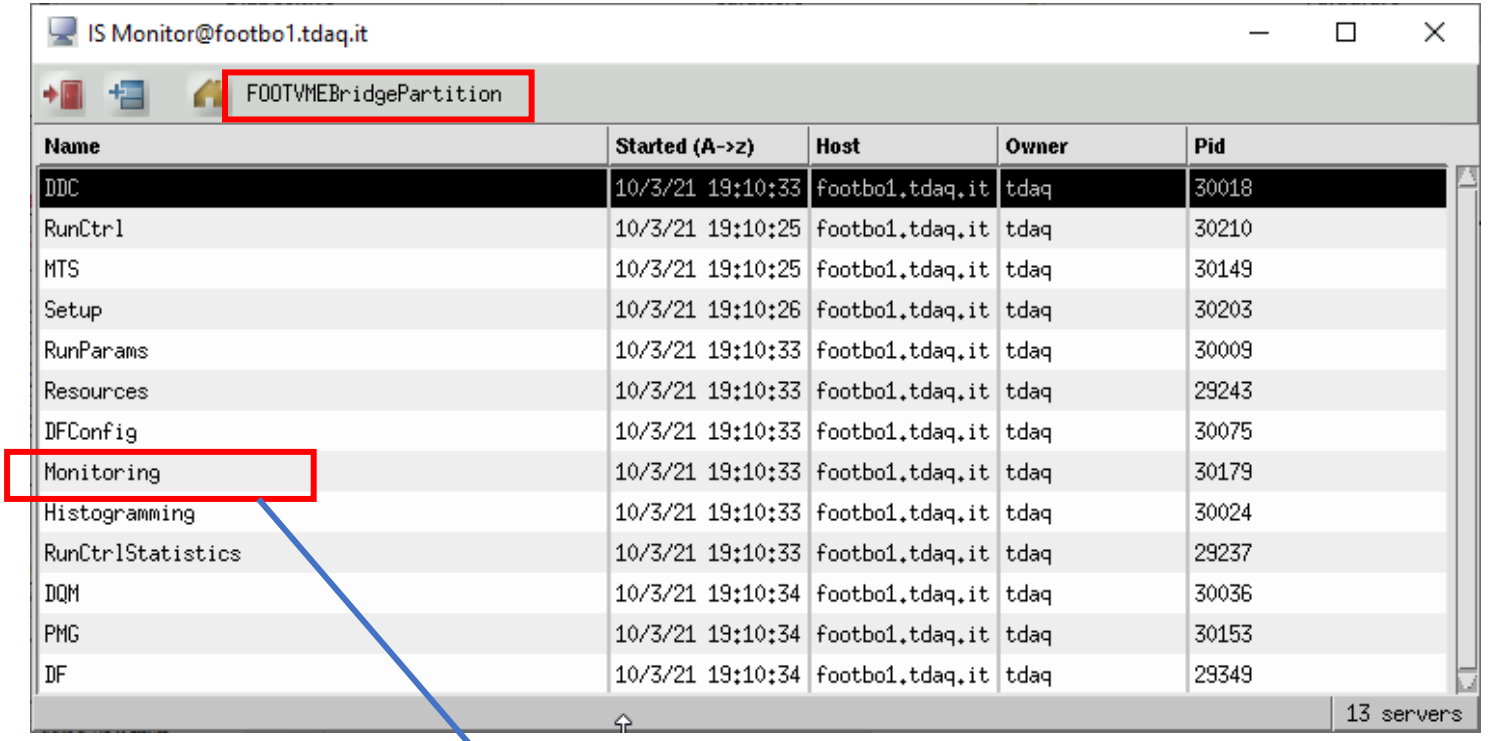
```
class ModuleDE0InfoNamed : public ISNamedInfo {
public:

  std::string     BoardName;      // BoardName
  unsigned int    PortNoSlow;     // Remote port number
  std::string     HostName;       // Host name of the remote server
  unsigned int    FWversion;      // Version of the firmware of the DE0 module




};
```

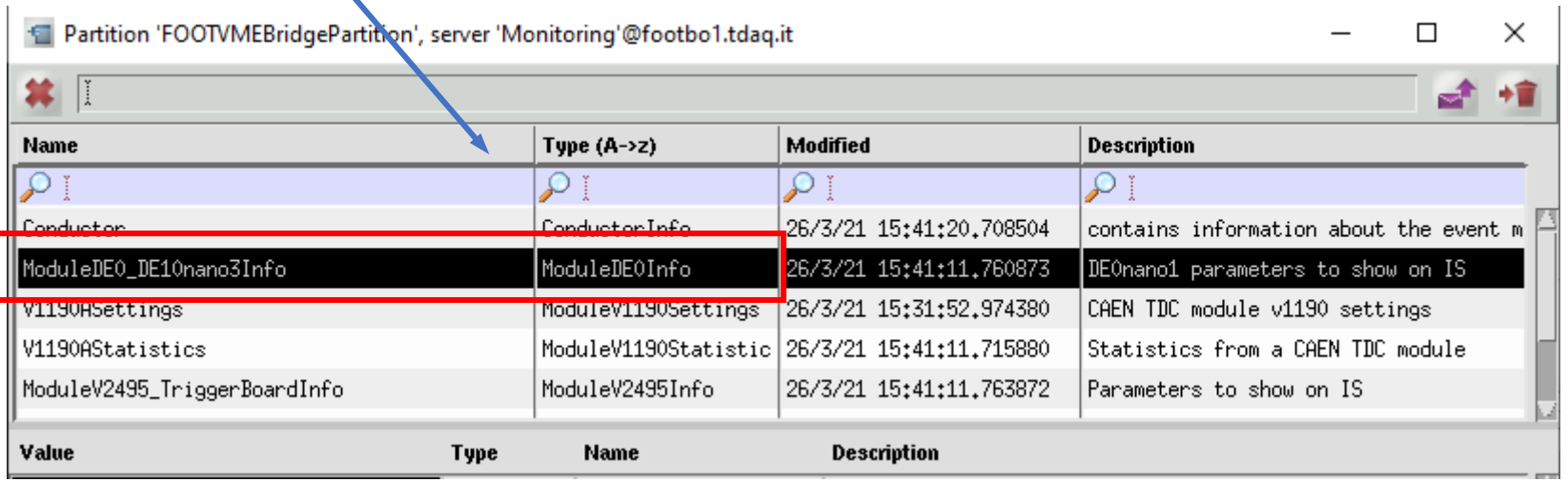
Instances of this class will store data and will be distributed via CORBA

Readout Module Publish()

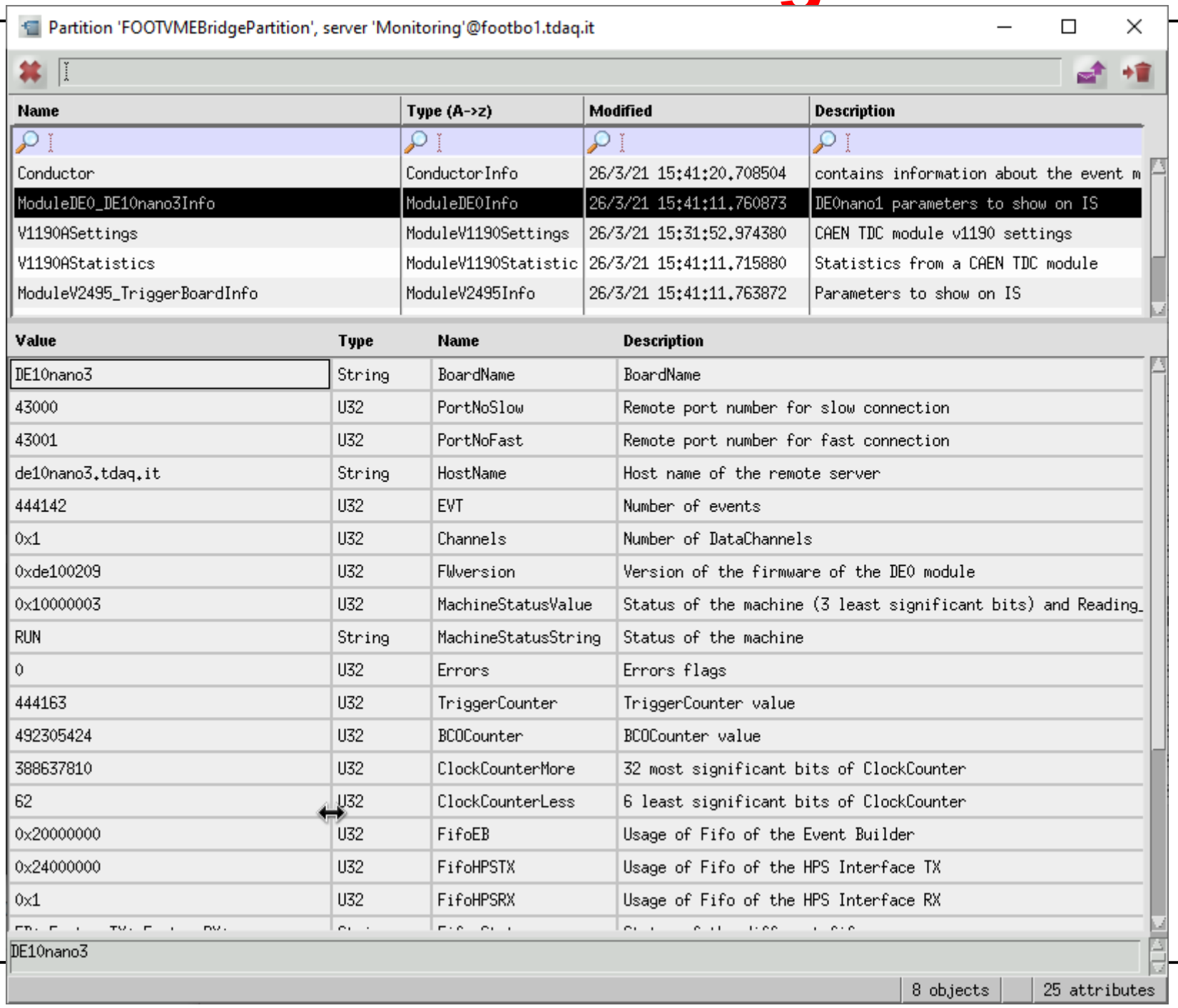
```
#include "rcd_Remote/ModuleDE0InfoNamed.h"

void ModuleDE0::publish(){

  std::string ModName="ModuleDE0_"+m_boardName+"Info";
  const std::string entryname = "Monitoring." + ModName;
  ModuleDE0InfoNamed is_entry(m_ipcpartition, entryname);

  is_entry.BoardName  = m_boardName;
  is_entry.PortNoSlow = m_portno;
  is_entry.HostName   = m_servName;
  is_entry.FWversion   = m_infoToPublish.at(0);  // value of register 16




};
```

On IS Monitoring .... $ is_monitor





On IS Monitoring ….



Online Histograms and Presenter

It is possible to define also histograms to publish periodically. Histograms are just a more complex CORBA objects that can be shipped around. In the framework they are mapped to ROOT histograms.

Histograms are stored locally as **objects** in **named** directories inside IS servers

ON IS:

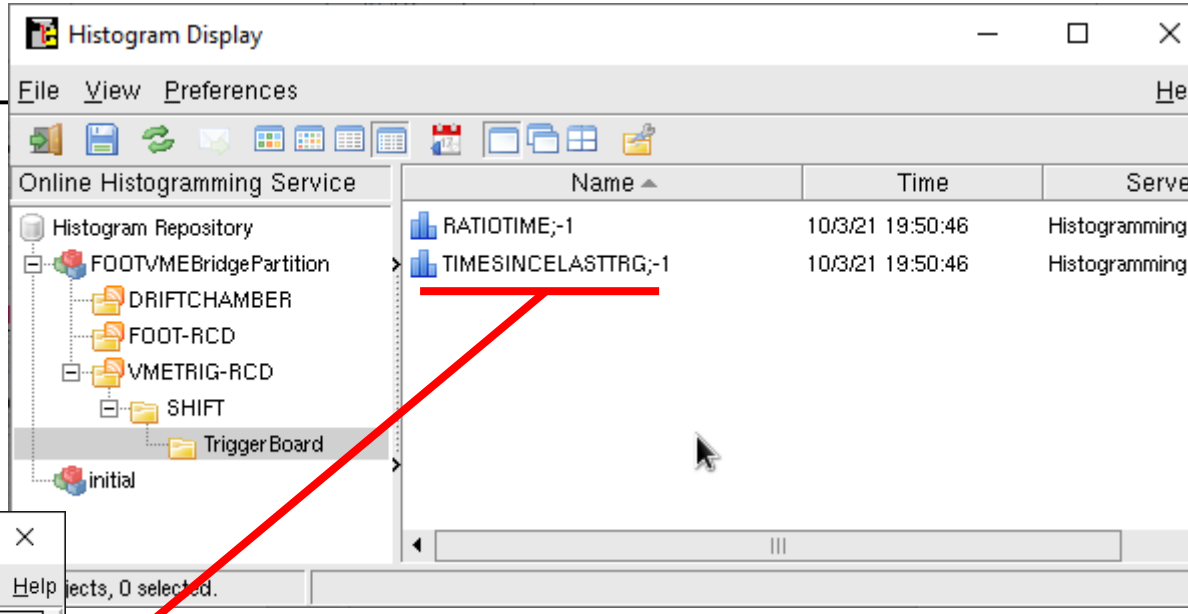
Readout Module V2495

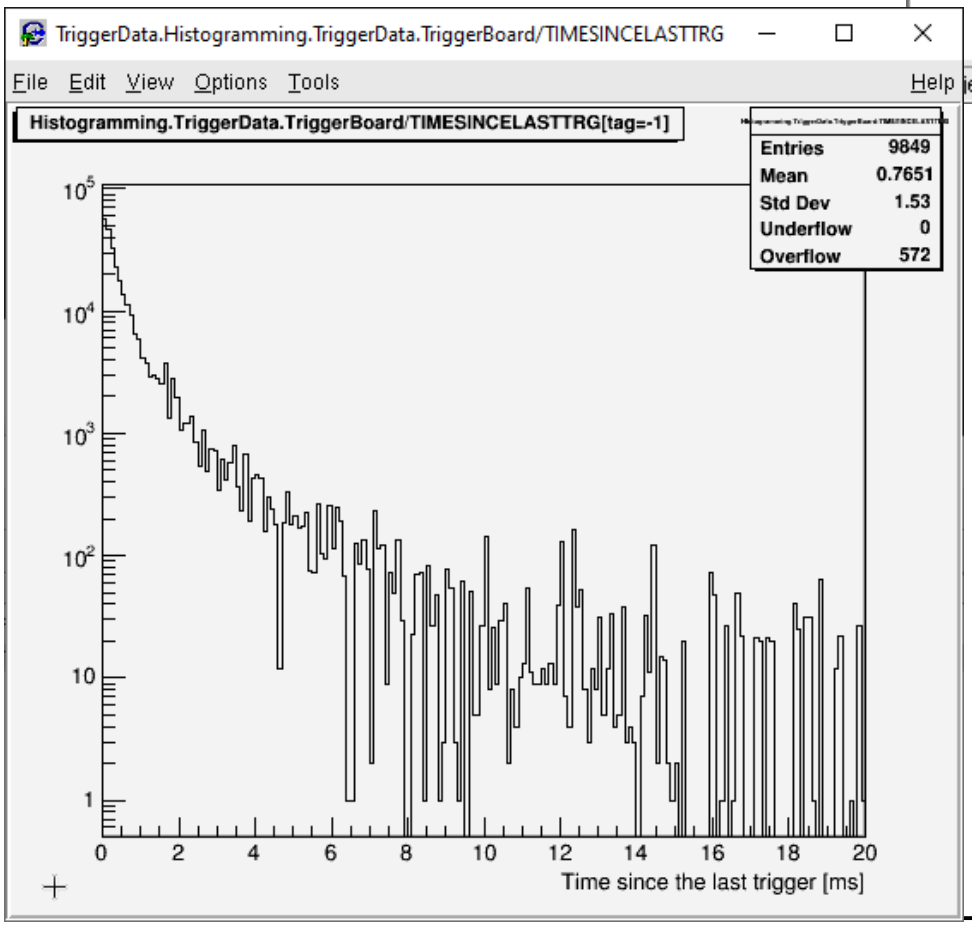
```
OHRootProvider  * m_oh_provider;      ← helper object
TH1D * m_htime;                       ← the actual ROOT histogram
```

OHP: presenter

Histograms are stored in directories

For reference, there is a group for each RCD





Data with the beam simulator:
- random distance between events.
- the “Effective” rate can be extracted from the mean time difference

All histograms can be saved at the end of a run

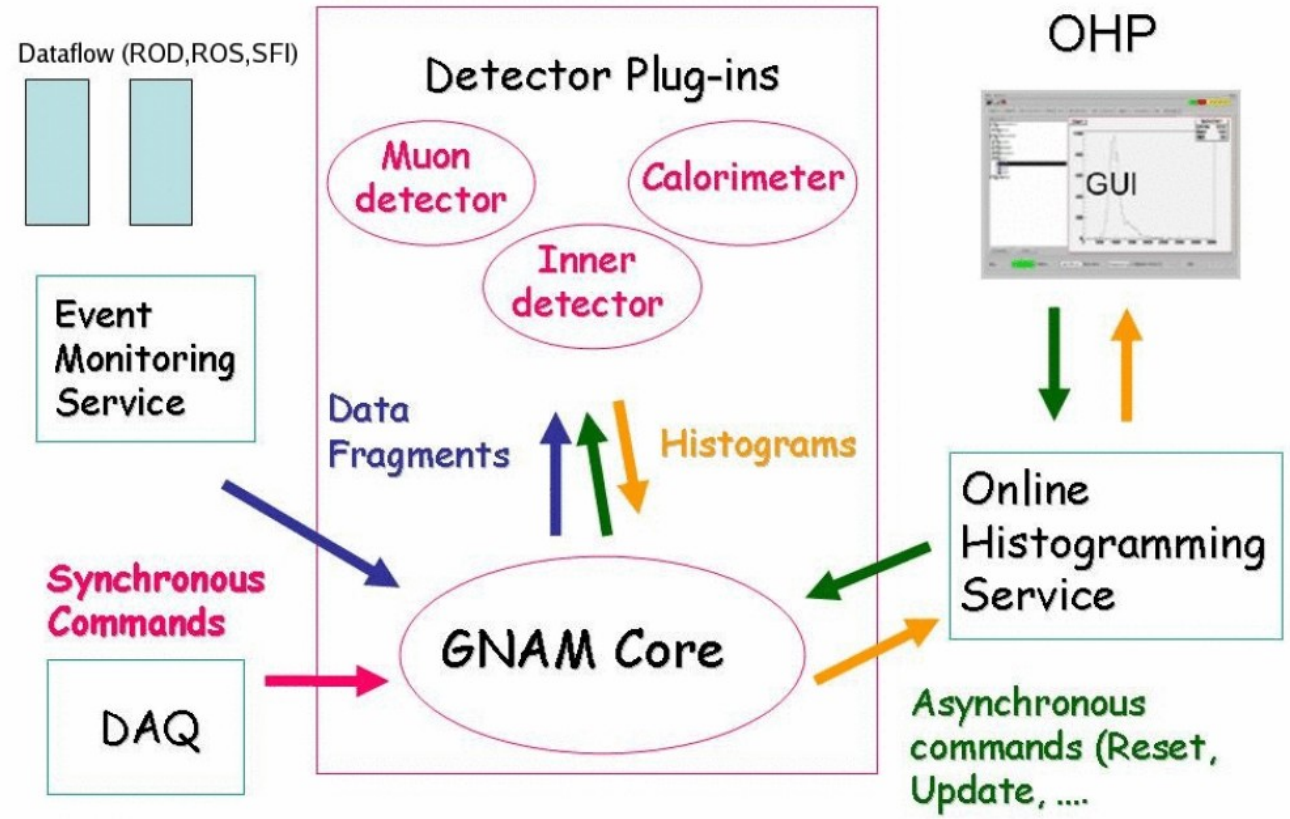
Pay attention to...

- Histograms are published by the Readout Modules publish() method.
- The DataChannel::getNextFragment is the only method that actually sees the data… but it is **time-critical**! Too much work here means a decrease of the DAQ rate!

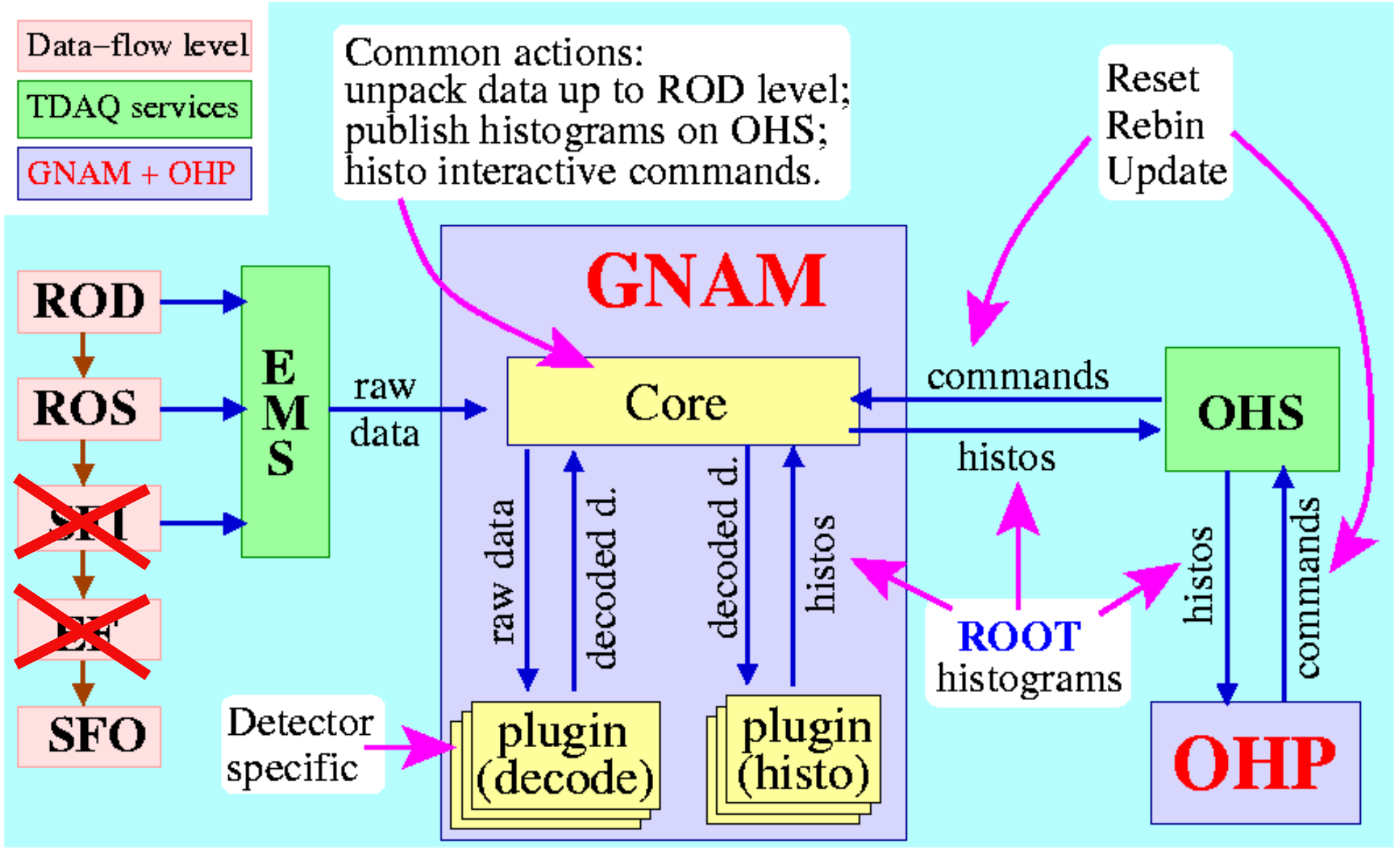
- Usually there is a quick filling of a data vector in the DataChannel::getNextFragment and from time to time (every 10 s) this is shipped to the Readout Module where it is used to fill an actual histogram.
- All data fragments can produce entries.
- Mind that the publish() and the getNextFragment() are running in separate threads.

GNAM

- A complementary approach to monitor all data from single readout modules is provided by GNAM: a full-event monitoring sampler



GNAM + OHP monitoring chain



GNAM in Foot

- GNAM is a **Controllable Application** with capability to have generic plug-ins

- In FOOT it is composed by just **three** elements:
  - GnamFootEvent
    - Storage class where to save data to be inserted in histos
  - GnamDecode
    - FOOT Stream decoding class: it fills the GnamFootEvent
  - GnamHisto
    - Creates and publish histos
    - Uses data in the GnamFootEvents to fill histos

GNAM Usage

- Its a monitoring tool: not time critical!
- It is a sampler of events, but it sees the full FOOT event
- You can monitor deeply inside a detector
- You can correlate different detectors
- We use it at least to check for time correlations

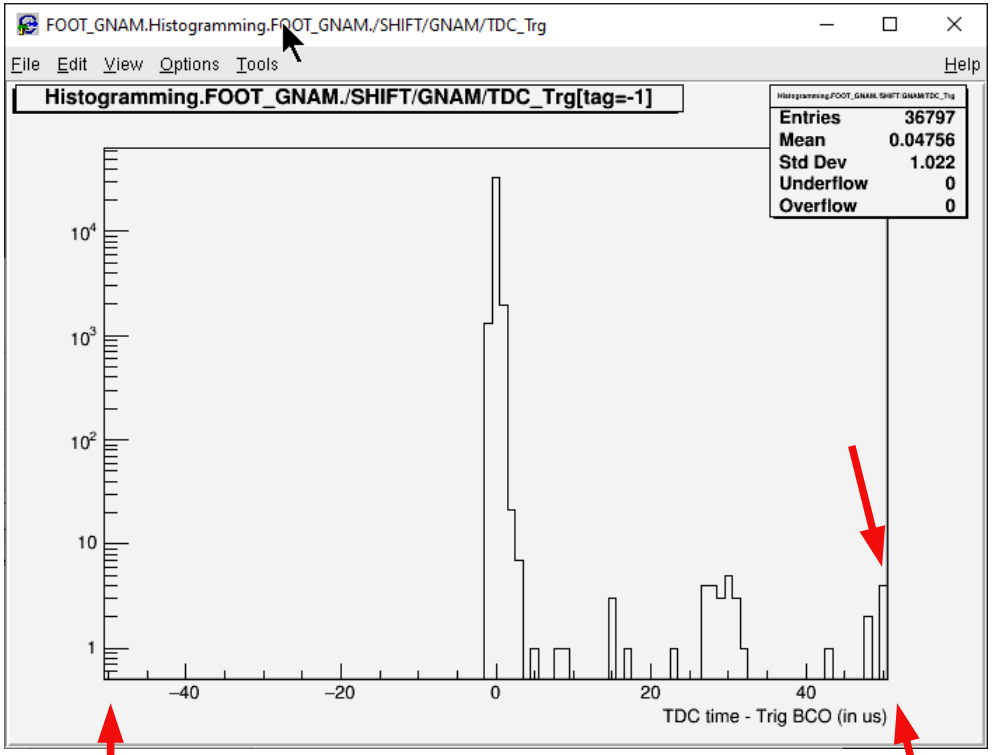
- In principle:
- You can search for tracks, interactions,fragmentations and make a full event reconstruction

GNAM Esamples

- Are the different detectors time aligned ?

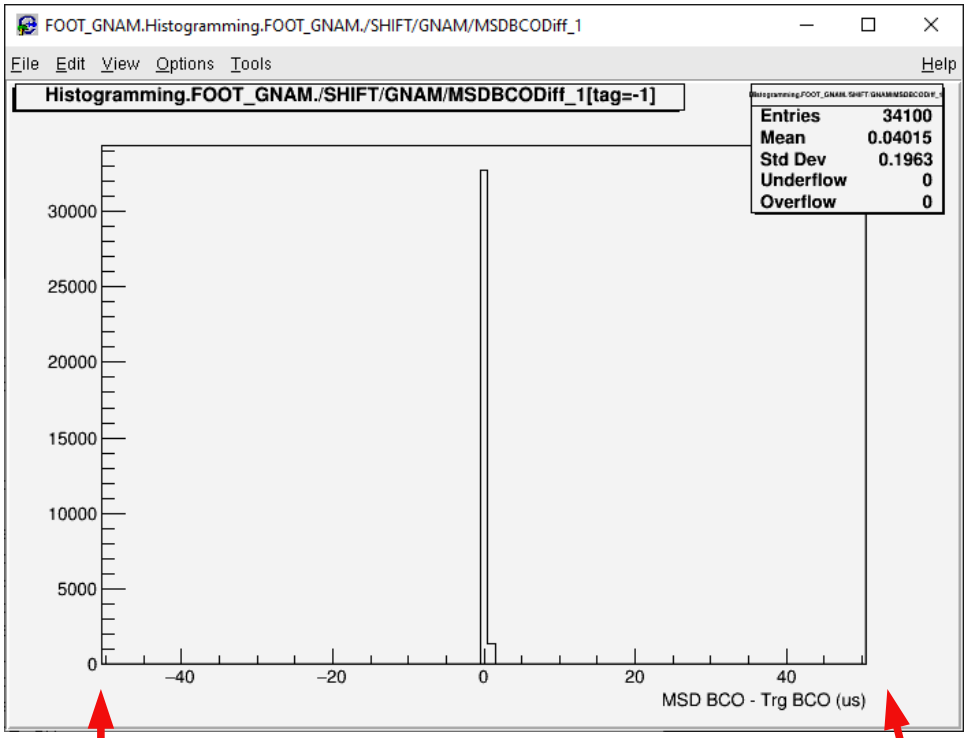
V2495 Makes the reference. TimeStamp clock at 1 MHz

TDC time tag – TS time difference
(wrt ev #0)



First & last bins are Under – Over flow bins

MSD: TS MSD – TS Trig



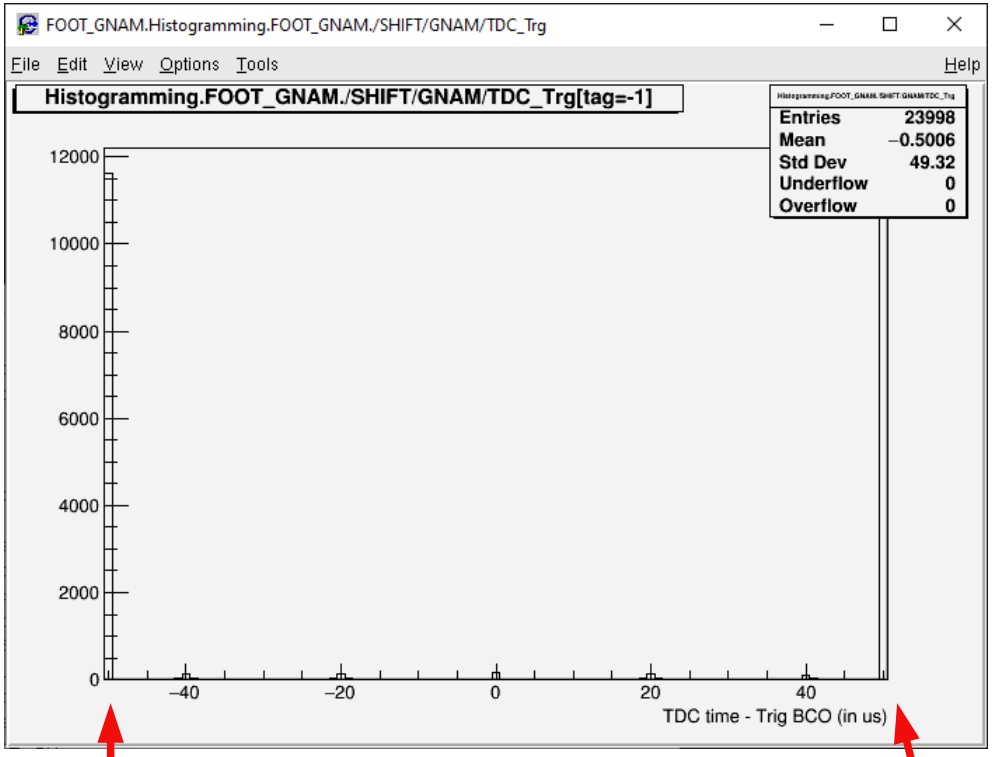
First & last bins are Under – Over flow bins

GNAM Esamples

- How would **you** judge these plots ??

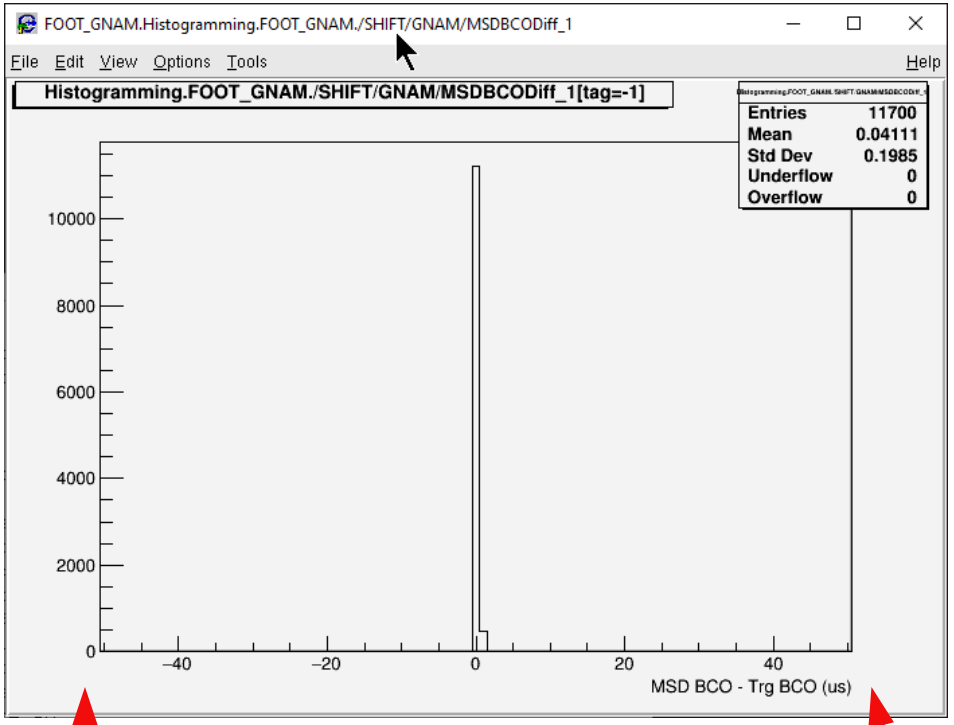
V2495 Makes the reference. TimeStamp clock at 1 MHz

TDC time tag – TS time difference
(wrt ev #0)



First & last bins are Under – Over flow bins

MSD: TS MSD – TS Trig



First & last bins are Under – Over flow bins

GNAM Status

- Gnam is the least advanced of all the infrastructure
- V2495 and TDC are there in a preliminary way
- VTX, MSD, TW and CALO have to be implemented
- “Simple” distributions are needed there

MYSQL

- Free and fast relational Data Base
- Used to store “low volume data” such as:
  - Detector settings
  - Running conditions
  - End of run counters
  -

Tables in database “tdaq”

How to access:

```
$ mysql -u tdaq -h footbo1 -p
Enter password:

Welcome to the MySQL monitor.
Your MySQL connection id is 642
Server version: 5.7.11 distribution



mysql> use tdaq;

mysql> show tables;
```

```
+----------------------+
| Tables_in_tdaq       |
+----------------------+
| RUNNUMBER            |
| V1190Config          |
| V2495Counters        |
| V2495Trigger         |
| V895                 |
| WDBoardCounters      |
| WDTriggerBoard       |
| WDTriggerConfig      |
+----------------------+
```

BM Configuration

```
mysql> select * from V1190Config where run>2210 and run<2213;
+----+------+----------+-------------+--------------+--------------+-….-+-------------+---------------+
| ID | run  | boardname | MatchingMode | TriggerOffset | TriggerWidth | ... | FIFOEnabled | BusyAlmostFull |
+----+------+----------+-------------+--------------+--------------+-….-+-------------+---------------+
| 31 | 2211 | V1190TN  |           1 |          -20 |           80 |     |           0 |              1 |
| 32 | 2212 | V1190TN  |           1 |          -20 |           80 |     |           0 |              1 |
+----+------+----------+-------------+--------------+--------------+-...-+-------------+---------------+
```

```
mysql> select ID,run,boardname,thres0,thres1,thres15,majority from V895 where run>2210 and run<2213;
+-----+------+----------+--------+--------+---------+----------+
| ID  | run  | boardname | thres0 | thres1 | thres15 | majority |
+-----+------+----------+--------+--------+---------+----------+
| 191 | 2211 | V895_A   |     20 |     20 |      20 |       56 |
| 192 | 2211 | V895_B   |     20 |     20 |      20 |       56 |
| 193 | 2211 | V895_C   |     20 |     20 |     100 |       56 |
| 194 | 2212 | V895_A   |     20 |     20 |      20 |       56 |
| 195 | 2212 | V895_B   |     20 |     20 |      20 |       56 |
| 196 | 2212 | V895_C   |     20 |     20 |     100 |       56 |
+-----+------+----------+--------+--------+---------+----------+
```

High Voltages were missing!!

End-Of-Run counters

```
mysql> select run,daqevt,TRGg,CTg,CTb from V2495Counters where run>3327;
+------+---------+---------+------------+------------+
| run  | daqevt  | TRGg    | LiveTime   |Elapsed time|
+------+---------+---------+------------+------------+
| 3328 |    1353 |    5984 |    5922364 |   61646416 |
| 3329 |     857 |    5481 |    5424160 |    6543030 |
| 3334 |  458829 |  458761 |  347942817 |  465618725 |
| 3335 | 2966122 | 2966525 |  505081671 |  803361892 |
| 3336 | 2574691 | 2575161 |  300641985 |  631528129 |
| 3337 |       0 |       0 |  137580831 |  137580930 |
| 3338 | 6252547 | 6252630 |  914887756 |  986614118 |
| 3339 |   10027 |   10105 |    4949189 |    5052460 |
| 3340 |   10074 |   10110 |    4951867 |    5055192 |
| 3341 | 3617077 | 3617432 | 1062109411 | 1186894407 |
+------+---------+---------+------------+------------+
```

Effective DAQ rates
DAQ efficiencies
“True Beam time”

Who fills the DB?

- **RUNNUMBER** table is filled by the central TDAQ
- Configuration information are filled by the **prepareForRun()** method in a readout module
- End-of-run counters are filled by the **StopDC()** method in a readout module

- If needed, a separate process (a Controllable Application) can read data from IS_servers and write complex data in a suitable DB table

Foot TDAQ components

- Hardware
  - Trigger Control
  - Beam Monitor
  - Wave DAQ (SC+TW+CALO)
  - Vertex DE10-Standard+PC
  - Inner Tracker
  - MicroStrip Detector
- Software
  - GNAM
  - Event Builder
  - EndOfRun Scripts
  - MYSQL, E-Logbook, GIT

Trigger control

- Central DAQV2495 Board
  - Firmware for the board! VHDL (Bagnari, Savarese, Bastia)
- TDAQ:
  - ModuleV2495 with a DataChannel
  - V2495TriggerIn (in VME-RCD)
  - V2495RemoteTriggerIn (in other RCD)

  Data provided:
  - In the Data stream (time of the event)
  - MYSQL: configuration and end-of-run counters
  - IS & OHP: run-time for checks

Beam Monitor

- V1x90 & V895 Boards
- TDAQ:
  - Module without a Data Channel V895: 16 channel discriminators (x3)
  - Module with a Data Channel V1x90: TDC
  -
- Data provided:
  - Time of wire signals in the data stream
  - MYSQL: configuration for each board
  - IS & OHP: run-time for checks

Micro Strip Detector

- 3 (or 6) DE10 nano boards
- TDAQ:
  - Module with a Remote Data Channel DE0Remote (x3, x6)
  -
- Data provided:
  - Strips hit in the data stream
  - MYSQL: *nothing at the moment*
  - IS & OHP: run-time for checks (To be completed)

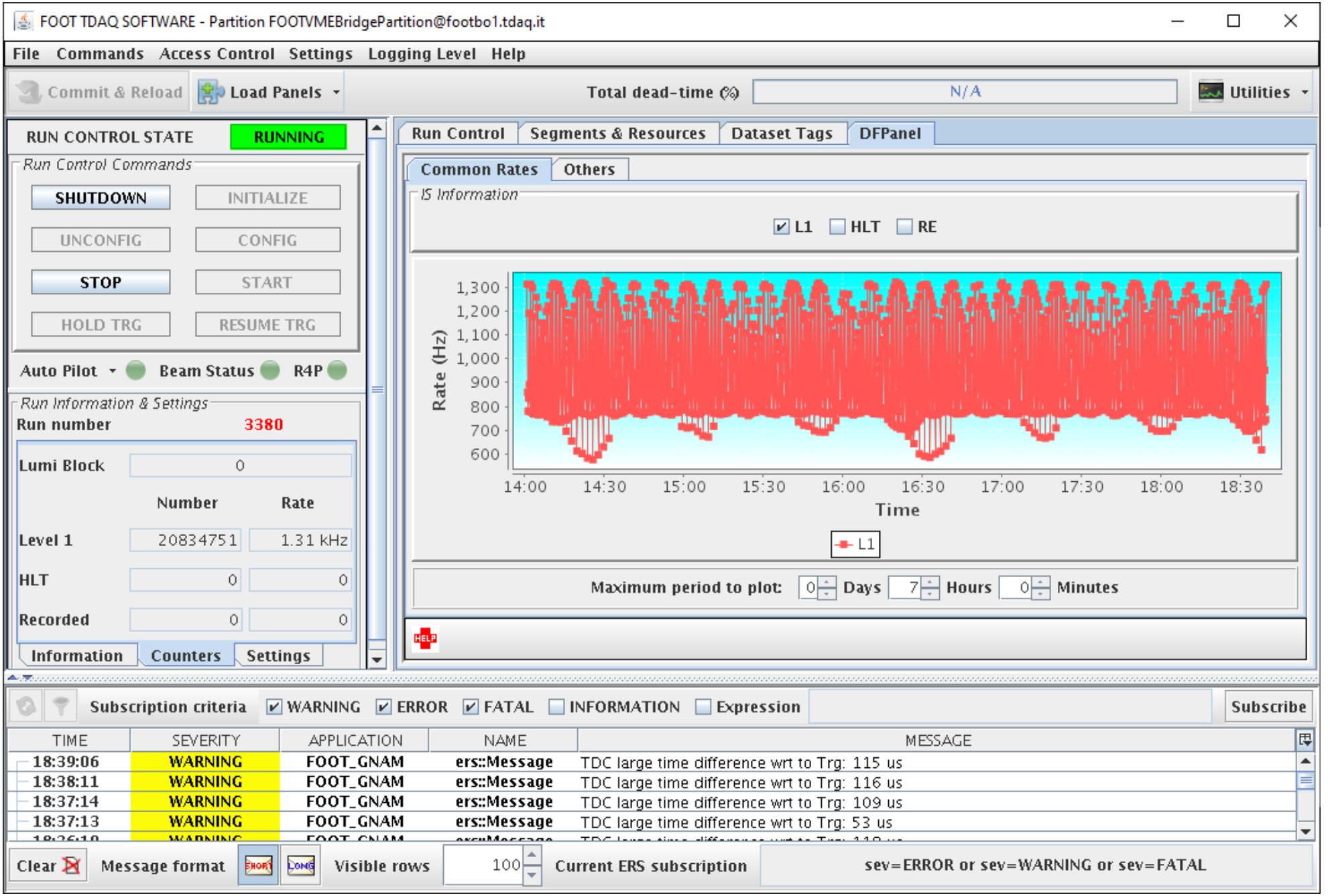
VTX & ITR

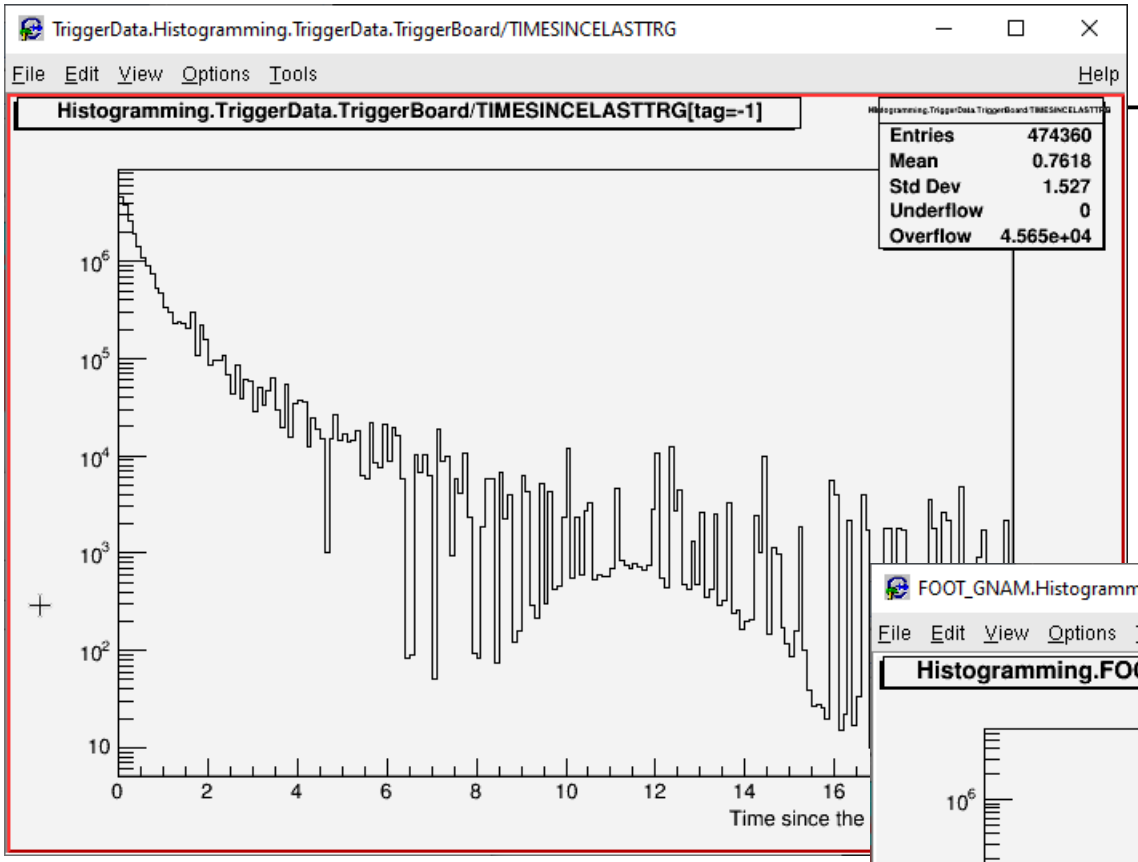
- 1 DE10 standard + 16 DE 10 nano boards
- Read by one or two independent PCs
- TDAQ:
  - ONE Module with a Remote Data Channel for VTX
  - ONE Module with a Remote Data Channel for ITR
- Data provided:
  - Pixel and Strip hits in the data stream
  - MYSQL: *nothing at the moment*
  - IS & OHP: run-time for checks (nothing at the moment)

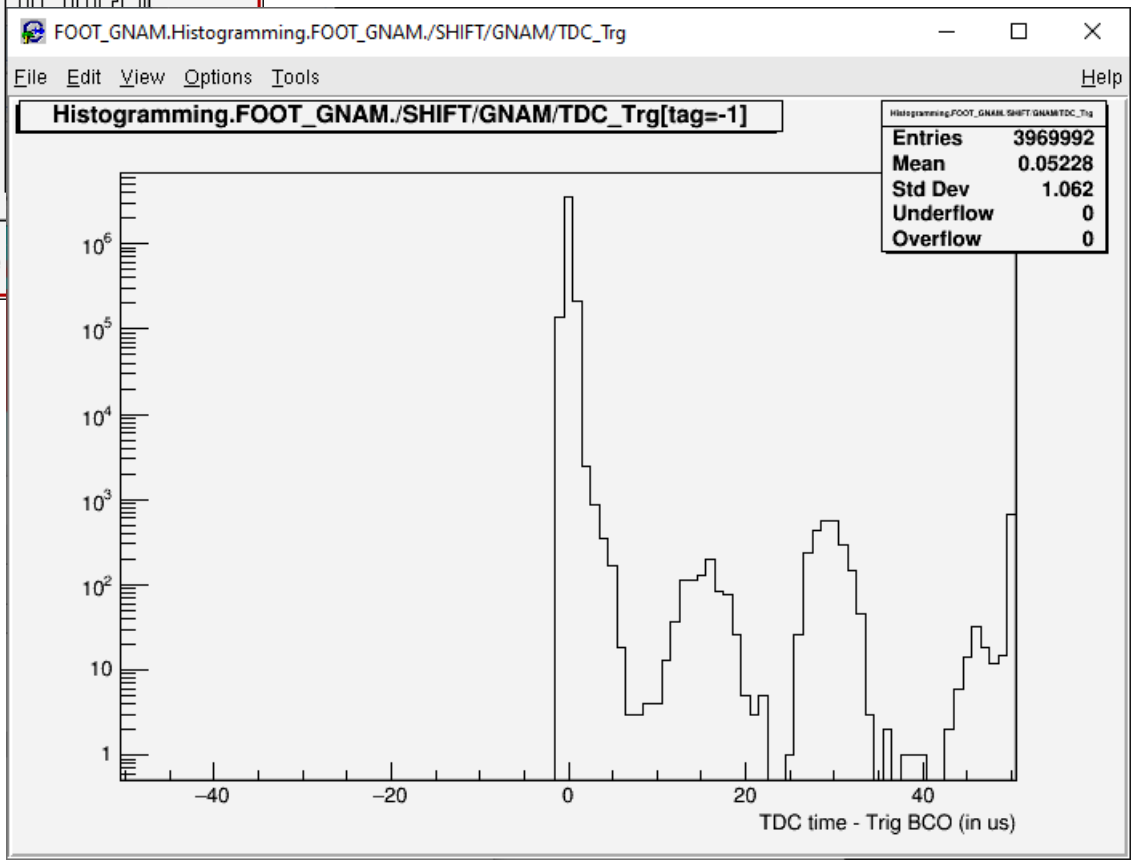
SC, TofWall, CALO, Neutrons

- 1 WaveDream acquisition system sending data to a custom PC
- Read by one or two independent PCs
- TDAQ:
  - ONE Module with a Remote Data Channel for WD

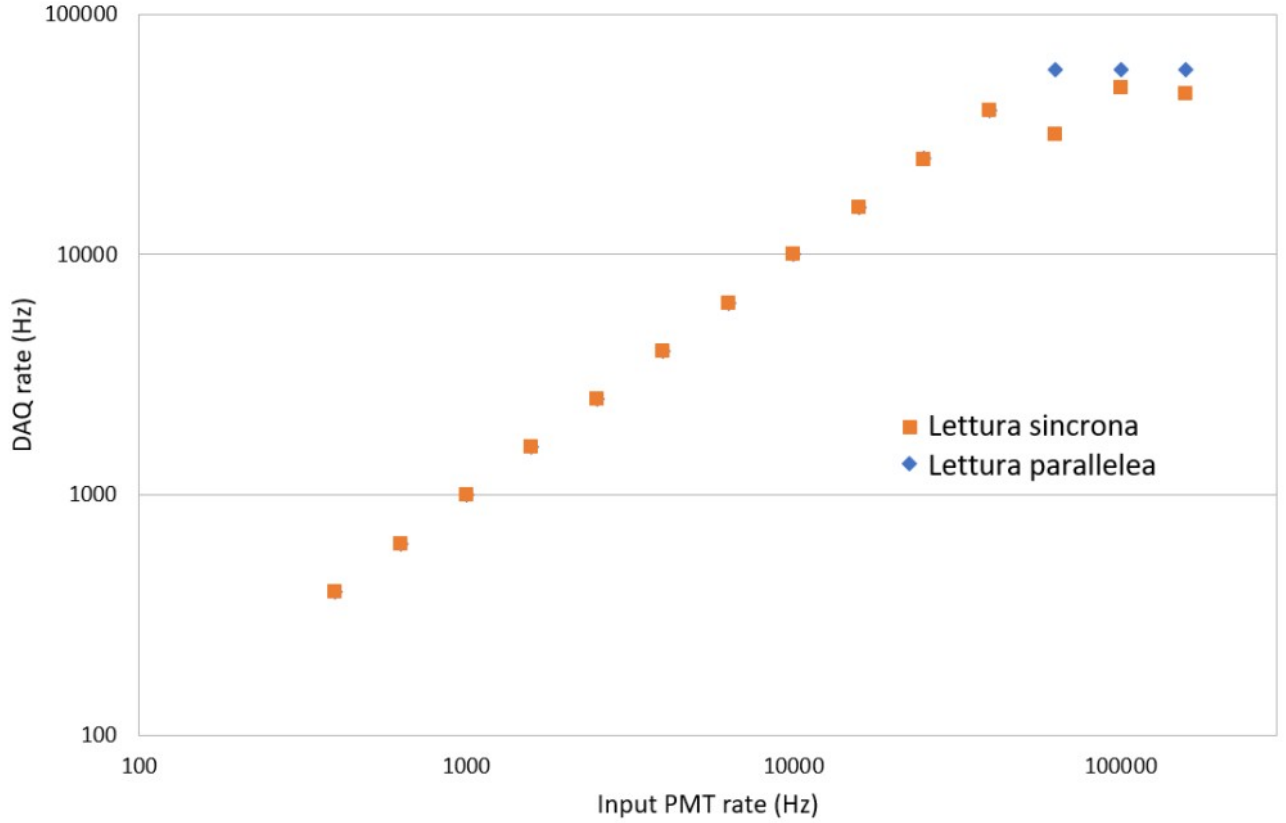
- Data provided:
  - waveforms in the data stream
  - Control data in the IS service
  - MYSQL: *nothing at the moment for CALO & Neutrons*
  - IS & OHP: run-time for checks (*nothing at the moment for CALO & n*)


FOOT TDAQ SOFTWARE - Partition FOOTVMEBridgePartition@footbo1.tdaq.it
File Commands Access Control Settings Logging Level Help
Commit & Reload
Load Panels
Total dead-time (%)
N/A
Utilities
RUN CONTROL STATE
RUNNING
Run Control Commands
SHUTDOWN
INITIALIZE
UNCONFIG
CONFIG
STOP
START
HOLD TRG
RESUME TRG
Auto Pilot
Beam Status
R4P
Run Information & Settings
Run number
3380
Lumi Block
0
Number
Rate
Level 1
20834751
1.31 kHz
HLT
0
0
Recorded
0
0
Information
Counters
Settings
Run Control
Segments & Resources
Dataset Tags
DFPanel
Common Rates
Others
IS Information
L1
HLT
RE
Rate (Hz)
1,300
1,200
1,100
1,000
900
800
700
600
14:00
14:30
15:00
15:30
16:00
16:30
17:00
17:30
18:00
18:30
Time
L1
Maximum period to plot:
0
Days
7
Hours
0
Minutes
HELP
Subscription criteria
WARNING
ERROR
FATAL
INFORMATION
Expression
Subscribe
TIME
SEVERITY
APPLICATION
NAME
MESSAGE
18:39:06 WARNING FOOT_GNAM ers::Message TDC large time difference wrt to Trg: 115 us
18:38:11 WARNING FOOT_GNAM ers::Message TDC large time difference wrt to Trg: 116 us
18:37:14 WARNING FOOT_GNAM ers::Message TDC large time difference wrt to Trg: 109 us
18:37:13 WARNING FOOT_GNAM ers::Message TDC large time difference wrt to Trg: 53 us
Clear
Message format
SHORT
LONG
Visible rows
100
Current ERS subscription
sev=ERROR or sev=WARNING or sev=FATAL


TriggerData.Histogramming.TriggerData.TriggerBoard/TIMESINCELASTTRG
File Edit View Options Tools Help
Histogramming.TriggerData.TriggerBoard/TIMESINCELASTTRG[tag=-1]
Entries 474360
Mean 0.7618
Std Dev 1.527
Underflow 0
Overflow 4.565e+04
Time since the

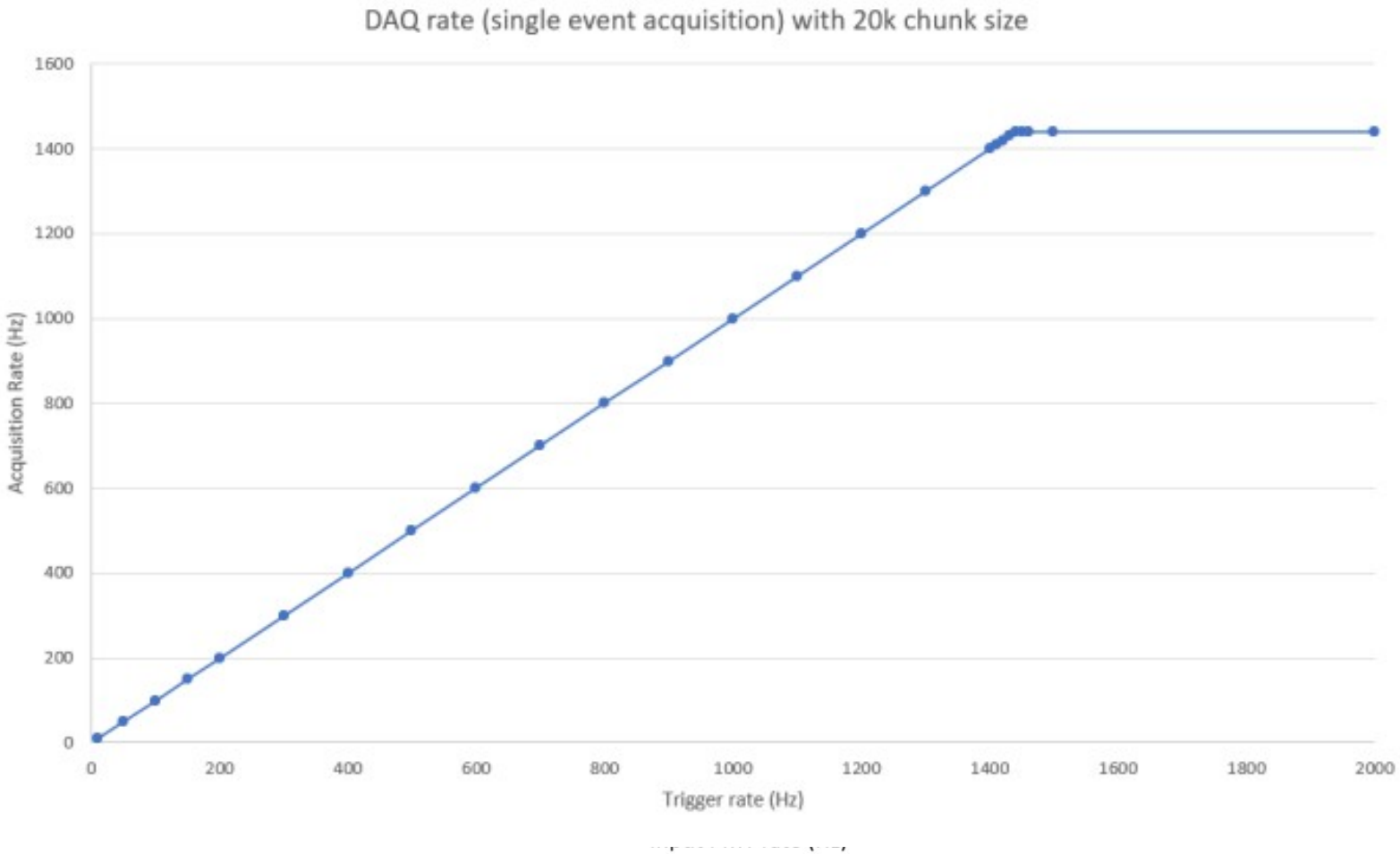

FOOT_GNAM.Histogramming.FOOT_GNAM./SHIFT/GNAM/TDC_Trg
File Edit View Options Tools Help
Histogramming.FOOT_GNAM./SHIFT/GNAM/TDC_Trg[tag=-1]
Entries 3969992
Mean 0.05228
Std Dev 1.062
Underflow 0
Overflow 0
TDC time - Trig BCO (in us)

Performance measurements



VME ONLY WITH SBC

Misura delle performances



DE10 with 1 event/TCPIP packet; data size 20kB (Savarese 2017).

Performance Measurements

In order to publish our DAQ system, when it will be completed we need:

Maximum steady rate
DAQ Rate vs Trigger rate
Live time vs elapsed time
Overall efficiency

Data volumes
Bandwidths at different points in the system
Writing capabilities (on SSD, on HD)

Monitoring information (volumes)
DB information (volumes)

"THAT'S THE END OF MY PRESENTATION. ANY QUESTIONS?"