Data AcQuisition System

Wojciech Byliński

for NA61/SHINE
Warsaw University of Technology

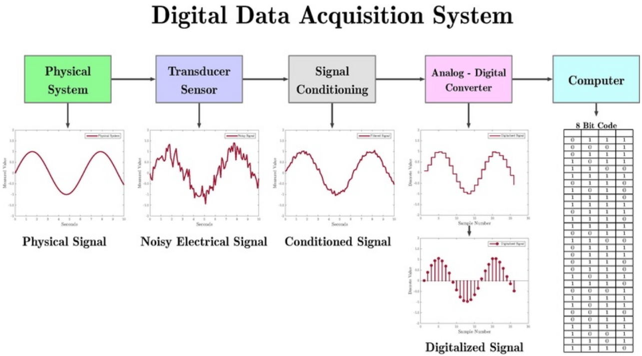
06.11.2020

Overview

Introduction

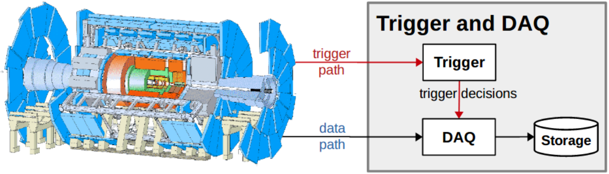
What is DAQ?

- Data AcQuisition (DAQ) is [Wikipedia]:
  - the process of sampling signals
  - that measure real world physical conditions
  - and converting the resulting samples into digital numeric values that can be manipulated by a PC

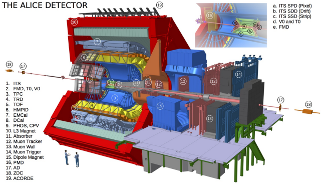


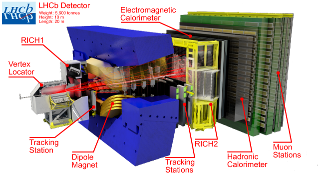
What is DAQ?

- Main role of DAQ:
  - process the signals generated in a detector
  - and saving the **interesting** information on a permanent storage



- Triggerless DAQ?

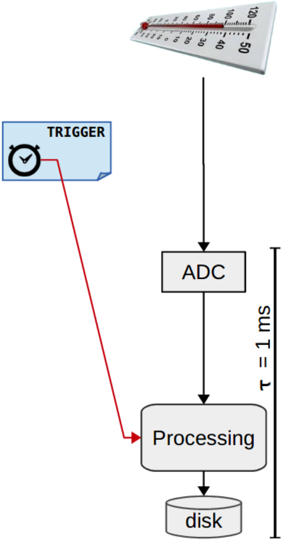




Basic concept

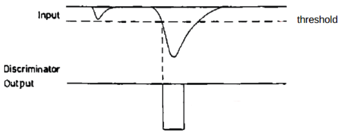
Periodic trigger

- Periodic measurements of the temperature
- System clearly limited by the time τ to process an "event"
- The DAQ maximum sustainable rate is simply the inverse of τ: $f_{max} = 1\text{kHz}$

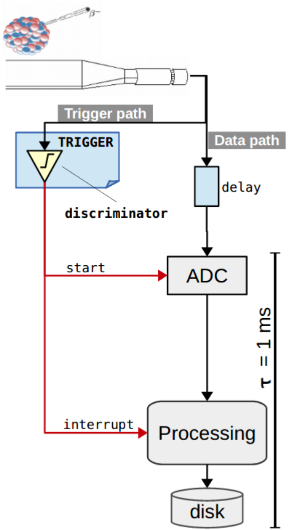


Realistic experiment

- Events asynchronous and unpredictable
- A physics trigger is needed:
  - Discriminator: generates an output digital signal if amplitude of the input pulse is greater than a given threshold

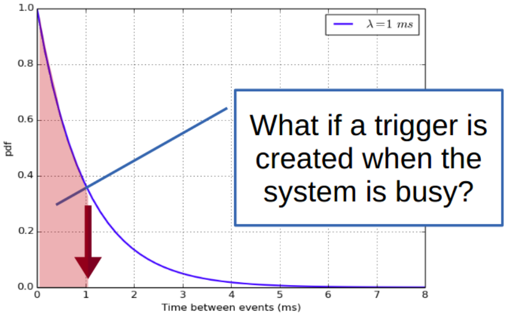


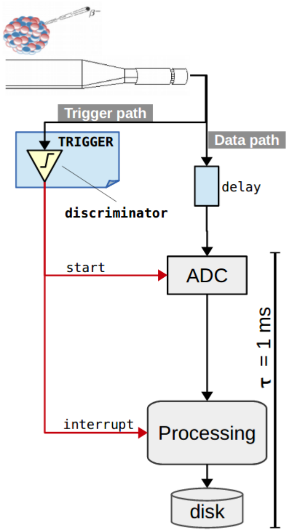
- delay introduced to compensate for the trigger latency
  - Signal split in trigger and data paths



Realistic experiment

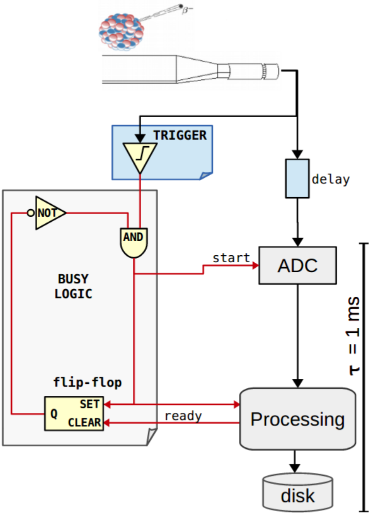
- Events asynchronous and unpredictable
- Lets assume:
  - physics rate f = 1kHz, i.e. $\lambda = 1\text{ms}^{-1}$





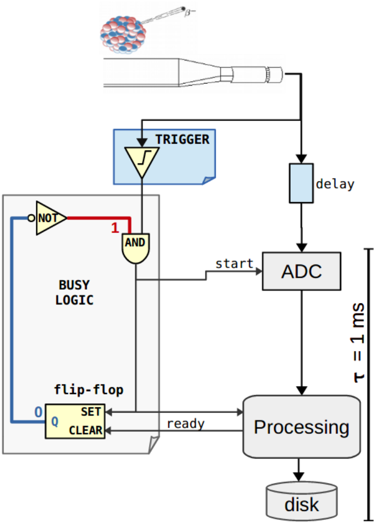
Busy logic

- Feedback mechanism to know if the data processing pipeline is free to process a new event
- A minimal busy logic can be implemented with:
  - an AND gate
  - a NOT gate
  - a flip-flop (circuit that changes state by signals applied to the control input)



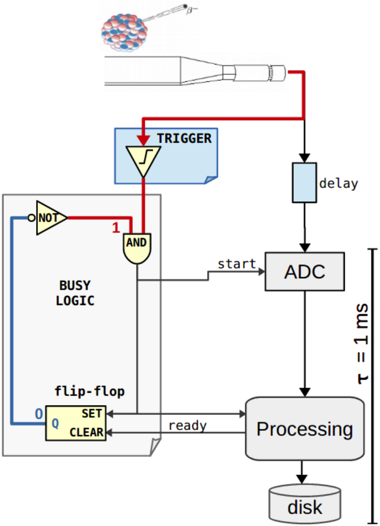
Busy logic

1. Start of new run: system is ready for new triggers
2. If a trigger arrives, the signal finds the AND gate open
3. The ADC is started, the processing is started, the flip-flop is flipped
4. One of the AND inputs is now steadily down (closed)
5. Any new trigger is inhibited by the AND gate (busy)
6. At the end of processing a ready signal is sent to the flip-flop
7. The system is ready for new trigger



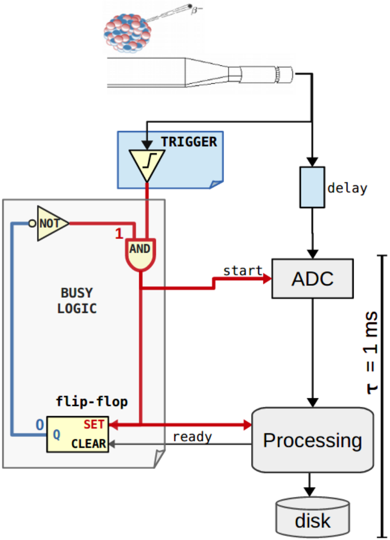
Busy logic

1. Start of new run: system is ready for new triggers
2. If a trigger arrives, the signal finds the AND gate open
3. The ADC is started, the processing is started, the flip-flop is flipped
4. One of the AND inputs is now steadily down (closed)
5. Any new trigger is inhibited by the AND gate (busy)
6. At the end of processing a ready signal is sent to the flip-flop
7. The system is ready for new trigger



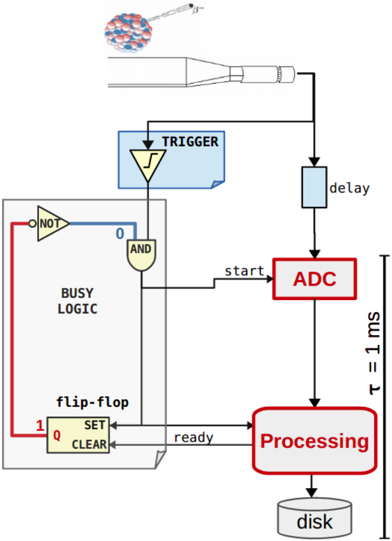
Busy logic

1. Start of new run: system is ready for new triggers
2. If a trigger arrives, the signal finds the AND gate open
3. The ADC is started, the processing is started, the flip-flop is flipped
4. One of the AND inputs is now steadily down (closed)
5. Any new trigger is inhibited by the AND gate (busy)
6. At the end of processing a ready signal is sent to the flip-flop
7. The system is ready for new trigger



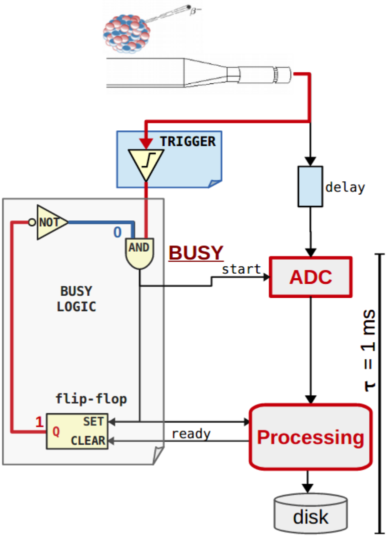
Busy logic

1. Start of new run: system is ready for new triggers
2. If a trigger arrives, the signal finds the AND gate open
3. The ADC is started, the processing is started, the flip-flop is flipped
4. One of the AND inputs is now steadily down (closed)
5. Any new trigger is inhibited by the AND gate (busy)
6. At the end of processing a ready signal is sent to the flip-flop
7. The system is ready for new trigger



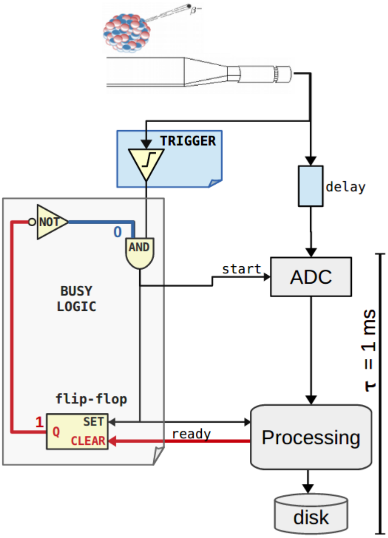
Busy logic

1. Start of new run: system is ready for new triggers
2. If a trigger arrives, the signal finds the AND gate open
3. The ADC is started, the processing is started, the flip-flop is flipped
4. One of the AND inputs is now steadily down (closed)
5. Any new trigger is inhibited by the AND gate (busy)
6. At the end of processing a ready signal is sent to the flip-flop
7. The system is ready for new trigger



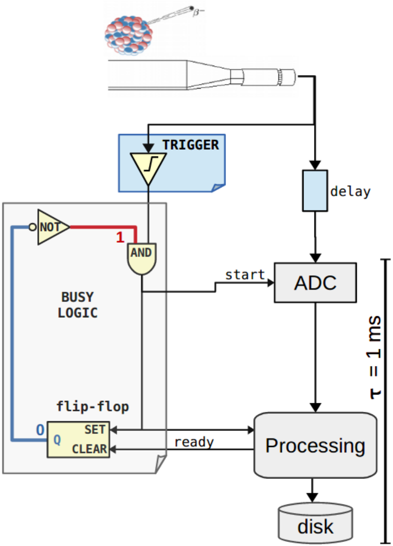
Busy logic

1. Start of new run: system is ready for new triggers
2. If a trigger arrives, the signal finds the AND gate open
3. The ADC is started, the processing is started, the flip-flop is flipped
4. One of the AND inputs is now steadily down (closed)
5. Any new trigger is inhibited by the AND gate (busy)
6. At the end of processing a ready signal is sent to the flip-flop
7. The system is ready for new trigger



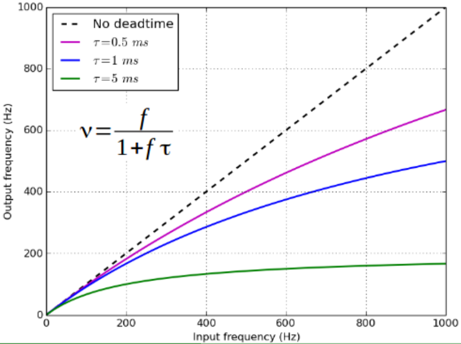
Busy logic

1. Start of new run: system is ready for new triggers
2. If a trigger arrives, the signal finds the AND gate open
3. The ADC is started, the processing is started, the flip-flop is flipped
4. One of the AND inputs is now steadily down (closed)
5. Any new trigger is inhibited by the AND gate (busy)
6. At the end of processing a ready signal is sent to the flip-flop
7. The system is ready for new trigger



Performance

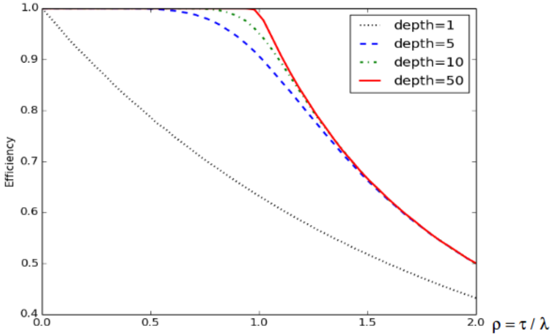
- Definitions:
  - f: average rate of physics (input)
  - γ: average rate of DAQ (output)
  - τ: deadtime, needed to process an event, without being able to handle other triggers
- Probabilities: $P[\text{busy}] = \gamma\tau$; $P[\text{free}] = 1 - \gamma\tau$
- Therefore: $\gamma = fP[\text{free}] \Rightarrow \gamma = f(1-\gamma\tau) \Rightarrow \gamma = \frac{f}{1+f\tau}$

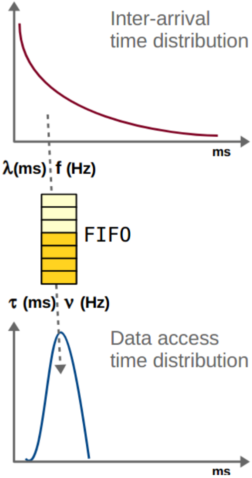


- $f = 1\text{kHz}$
- $\tau = 1\text{ms}$
- $\gamma = 500\text{Hz} \Rightarrow$ Efficiency = 50%
- To achieve 99% efficiency $\Rightarrow$ $\tau = 0.01\text{ms} \Rightarrow$ over-design the DAQ by factor of **100!**

De-randomization

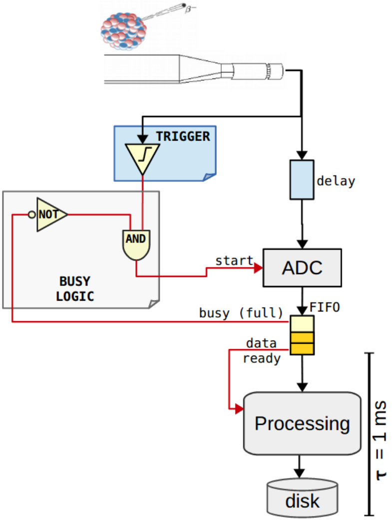
- What if we were able to make the system less dependent on the arrival time of our signals?
- Then we could ensure that events don't arrive when the system is busy – this is called de-randomization
- Can be achieved by buffering the data





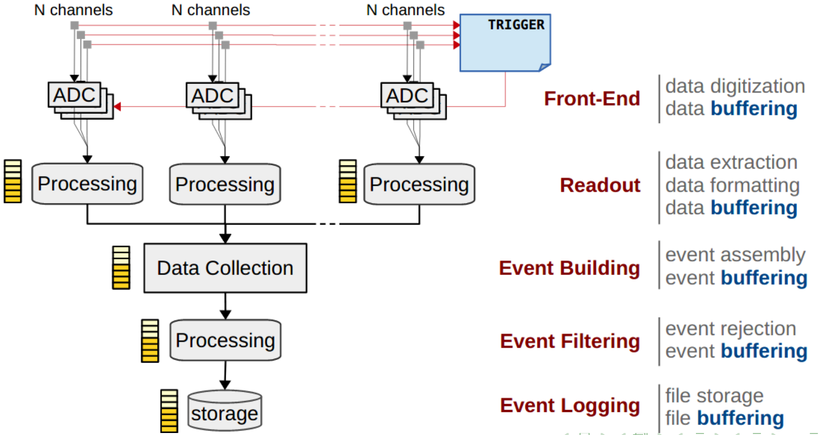
Final system

- Input fluctuations can be absorbed and smoothed by a queue
- Busy is now defined by the buffer occupancy



Big experiments

- Buffering usually needed at every level
- Basic concepts: Digitization, Latency, Deadtime, Busy, Back-pressure, De-randomization

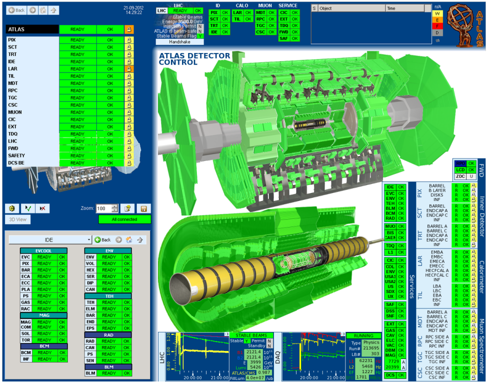


DAQ online software

DAQ Online Software

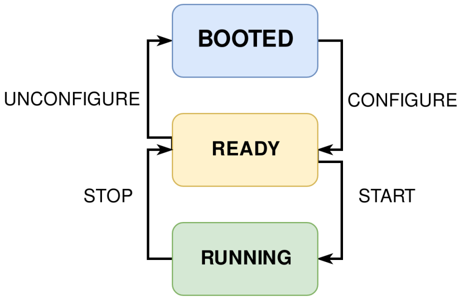
Main roles:

- configuration
- control
- monitoring



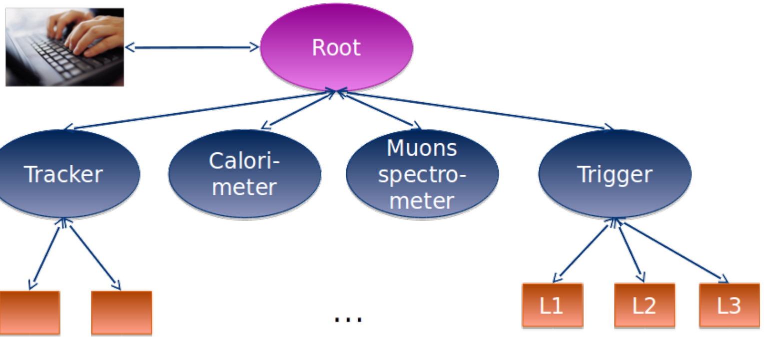
Finite State Machine

- Defines the behaviors of a system or a complex object, with a finite number of defined conditions or modes
- Transition between the states occurs only under strictly defined conditions



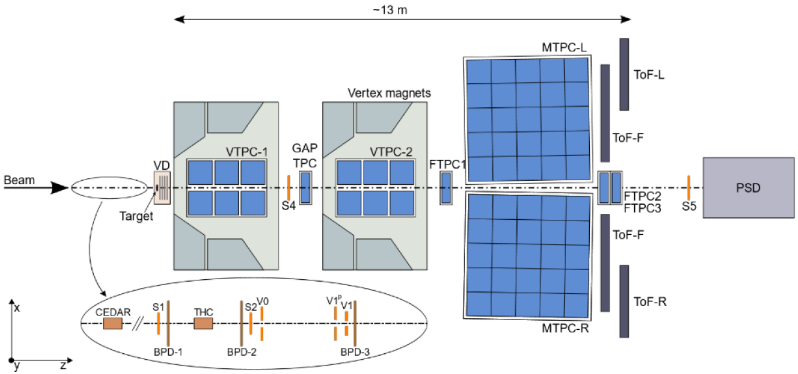
Run Control Tree

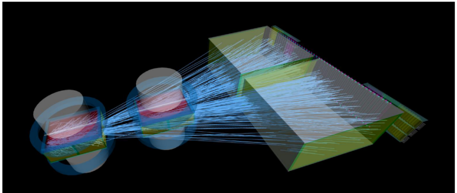
- Subdivision of the experiment into a tree, with well defined branches that may be acted upon independently
- The Run Control system provides a framework for applications, such that the mechanics of receiving/responding to commands is hidden



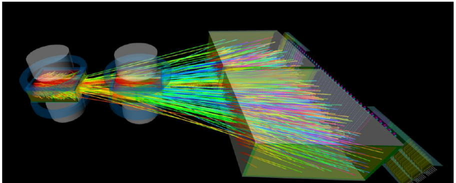
NA61/SHINE DAQ upgrade

NA61/SHINE detector



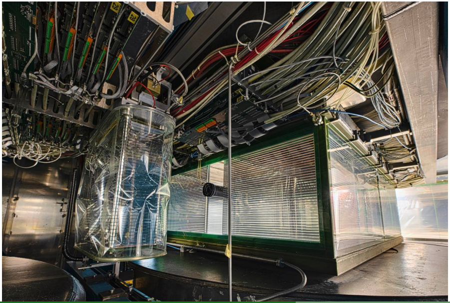


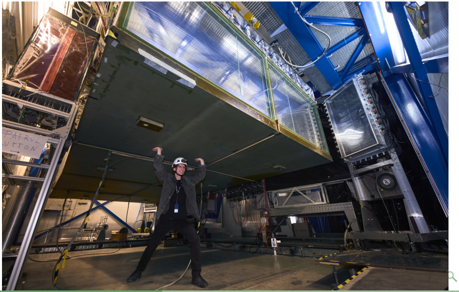
Pb+Pb at 150*A* GeV/*c* from 2016



Xe+La at 150*A* GeV/*c* from 2017

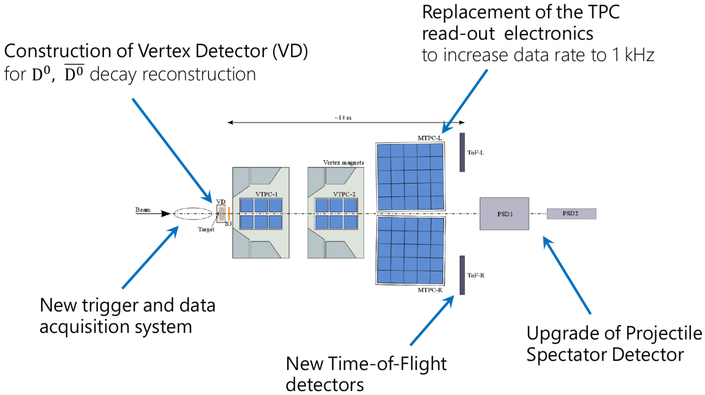
NA61/SHINE detector





NA61/SHINE upgrade

1. 10 fold increase of data taking rate up to 1 kHz
2. improvement of acceptance and efficiency of the Vertex Detector
3. improvement of radiation tolerance of the PSD hadron calorimeter
4. introduction of new TOF detector based on mRPC technology
5. replacement of old readout electronics based on CAMAC and FASTBUS standards



DAQ upgrade motivation

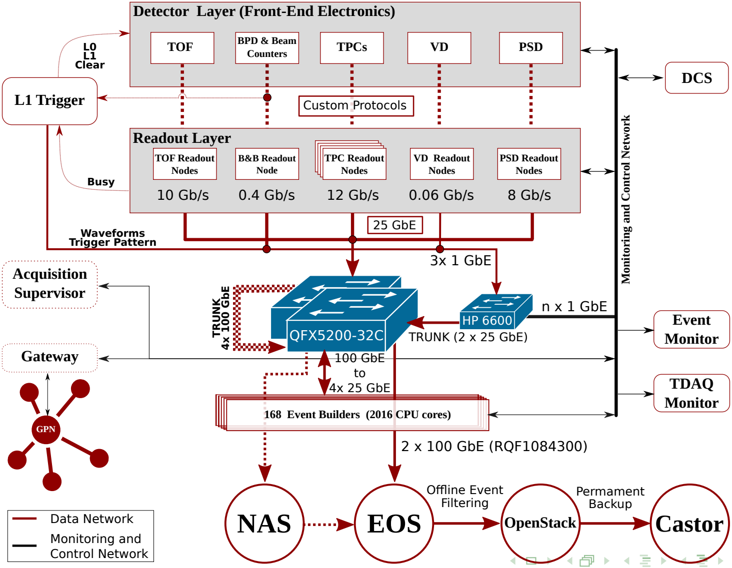
Motivation

- A key motivation for developing the development of new DAQ is to increase event rate from 80Hz to about 1KHz
- easy adding of the new sub-detectors

Challenges:

- huge data rates - TPC produces big amount of data
- online noise filtration needed

Data flow

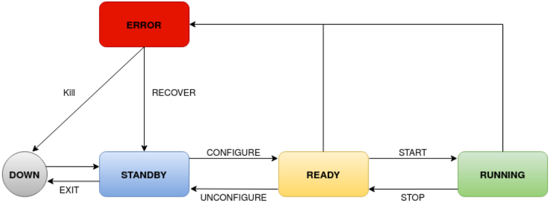

Detector Layer (Front-End Electronics)
TOF
BPD & Beam Counters
TPCs
VD
PSD
L0
L1
Clear
L1 Trigger
Custom Protocols
DCS
Readout Layer
TOF Readout Nodes
B&B Readout Node
TPC Readout Nodes
VD Readout Nodes
PSD Readout Nodes
10 Gb/s
0.4 Gb/s
12 Gb/s
0.06 Gb/s
8 Gb/s
Busy
25 GbE
Waveforms
Trigger Pattern
3x 1 GbE
Monitoring and Control Network
Acquisition Supervisor
TRUNK
4x 100 GbE
QFX5200-32C
HP 6600
n x 1 GbE
TRUNK (2 x 25 GbE)
Event Monitor
Gateway
100 GbE
to
4x 25 GbE
TDAQ Monitor
GPN
168 Event Builders (2016 CPU cores)
2 x 100 GbE (RQF1084300)
Offline Event Filtering
Permanent Backup
NAS
EOS
OpenStack
Castor
Data Network
Monitoring and Control Network

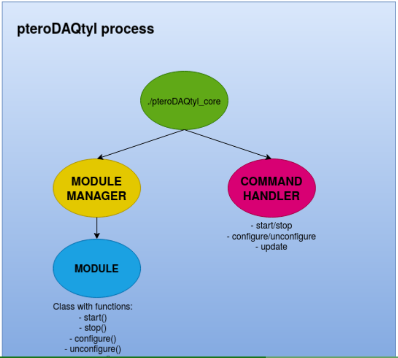
Software framework

pteroDAQtyl - a software framework enabling a uniform way of controlling and running the TDAQ processes

Design is based on **DAQling** project

pteroDAQtyl process

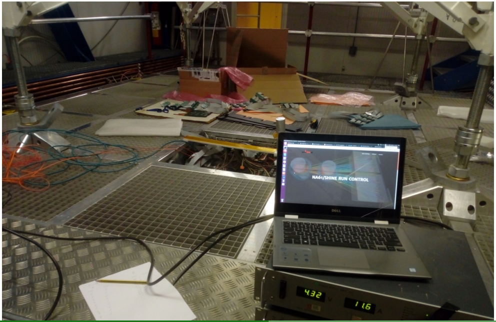


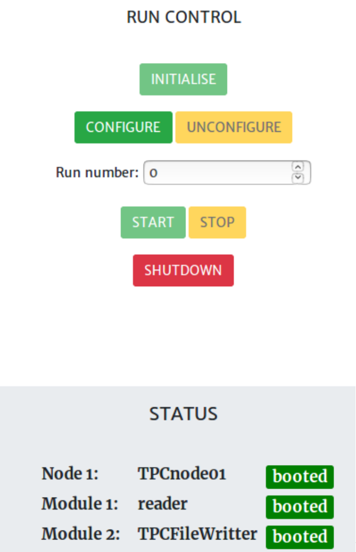


- ModuleManager can load a configurable number of Modules
- Module is a class inheriting from DAQModule class and defining the following functions: start()/stop(), configure()/unconfigure(), runner(), etc.
- Modules can communicate between each others (sending the sub-events) either by common events queue by passing the pointer to event object or by sending the event using ZeroMQ via DataManager.

pteroDAQtyl process control

- Python Supervisor wrapper (with prototype Web Application)
- JSON configuration (with prototype Web Application)
- JSON based communication

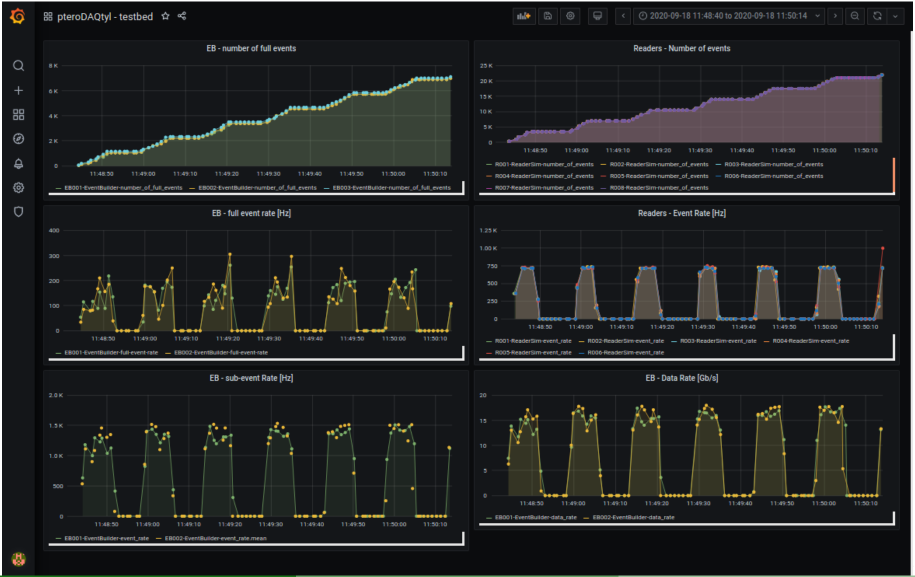




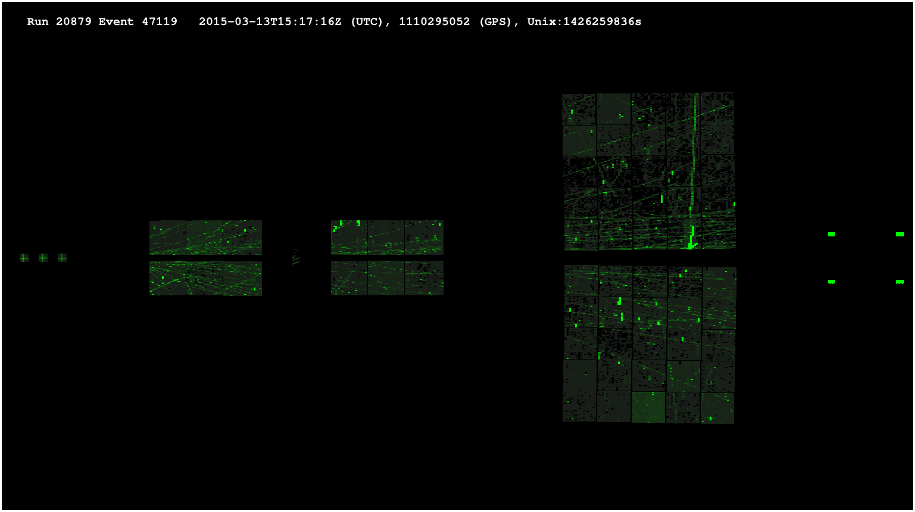
Monitoring

- Each Module can register any number of variables to MetricManager with a defined time interval
- With a given time interval, MetricManager sends the current values (or rate or mean) to the database (influxDB)
- Grafana used for vizualisation

Software tests



NA61/SHINE raw data monitor



Summary

- Basic concepts:
  - digitization, latency, deadtime, busy, de-randomization, back-pressure
- DAQ online software:
  - configuration
  - control
  - monitoring

Thank you!!!

wobrylin@cern.ch

Have a SHINY day!!! ;D

References

1. ISOTDAQ: https://indico.cern.ch/event/828931/
2. A. Negri "Introduction to Data AcQuisition", ISOTDAQ 2020
3. DAQling: https://gitlab.cern.ch/ep-dt-di/daq/daqling
4. Supervisor: http://supervisord.org/
5. InfluxDB: https://www.influxdata.com/
6. Grafana: https://grafana.com/