An Introduction to DAQ

Kristian Hahn – Northwestern

EDIT 2024, Fermilab

November 20, 2024

DAQ

“DAQ” = Data AcQuisition

- The medium between the physics and our experience of it

DAQ

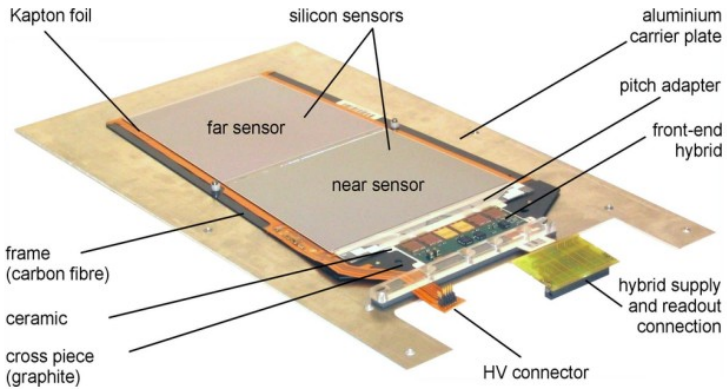
“DAQ” = Data AcQuisition

- The medium between the physics and our experience of it
- A catchall term with an expansive scope

DAQ

“DAQ” = Data AcQuisition

- The medium between the physics and our experience of it
- A catchall term with an expansive scope

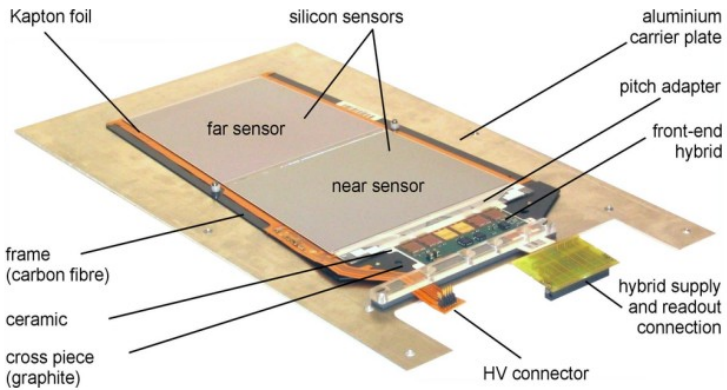


From frontend readout ...

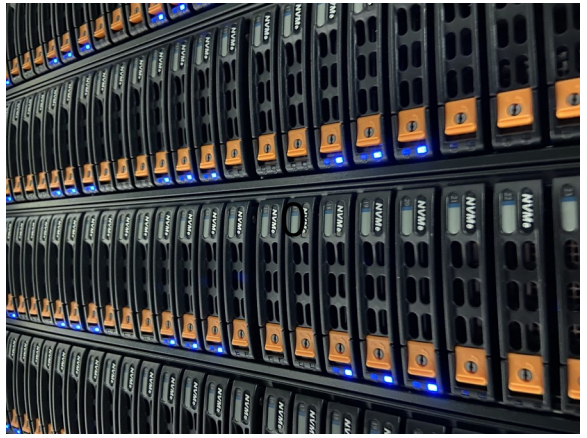
DAQ

“DAQ” = Data AcQuisition

- The medium between the physics and our experience of it
- A catchall term with an expansive scope



From frontend readout ...

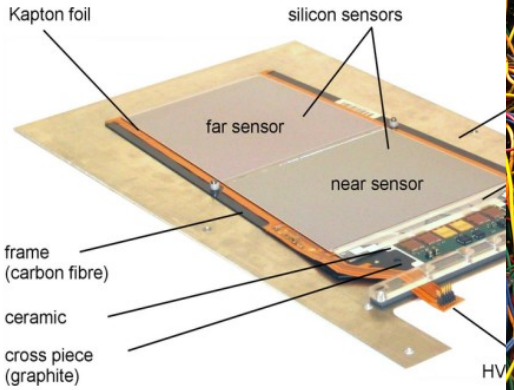


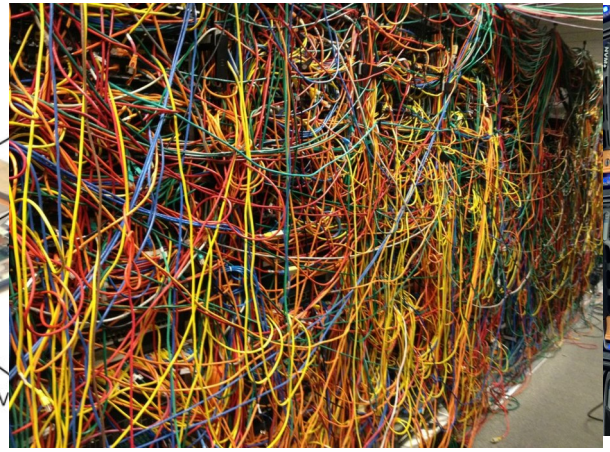
… to data on disk

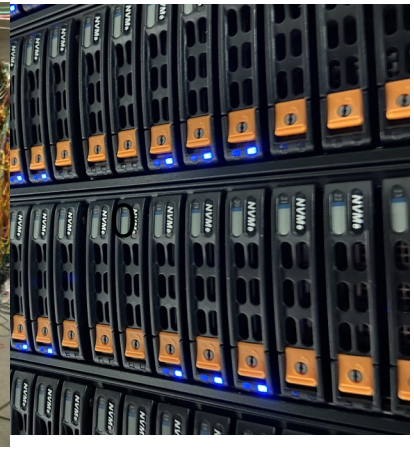
DAQ

“DAQ” = Data AcQuisition

- The medium between the physics and our experience of it
- A catchall term with an expansive scope







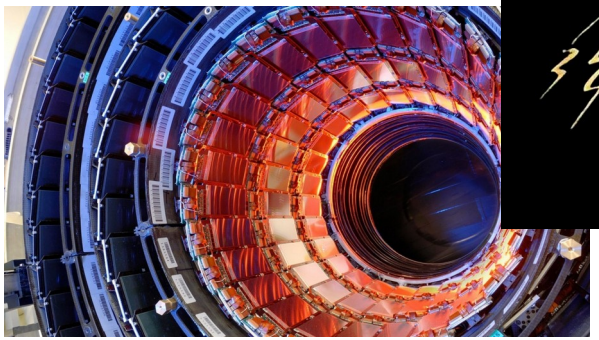
From frontend readout ... … to data on disk

and everything in between

DAQ

“DAQ” = Data AcQuisition

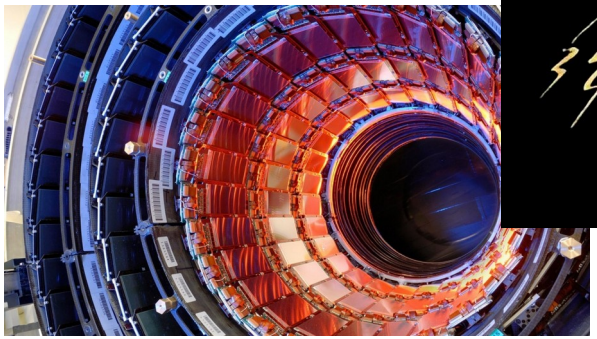
- The medium between the physics and our experience of it
- A catchall term with an expansive scope
- Transforming our collection of sensors …



DAQ

“DAQ” = Data AcQuisition

- The medium between the physics and our experience of it
- A catchall term with an expansive scope
- Transforming our collection of sensors …
  … into a functional scientific instrument!



DAQ Basics

Everyday DAQ

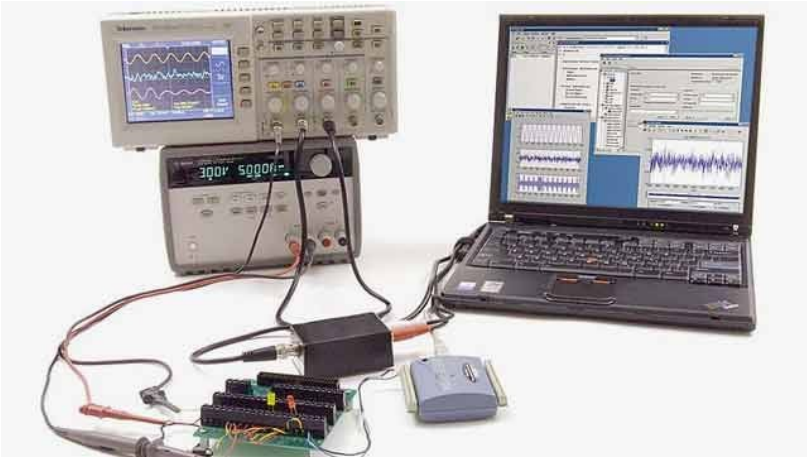
You're already familiar w/ DAQ!

- Analog signal sampling and digitization
- High speed data transmission
- DMA, bussed (USB, PCIe) communication and resource sharing
- Middleware (ie: drivers) for device control
- Software processing & data storage

More Sophisticated

Proprietary DAQ systems, eg:

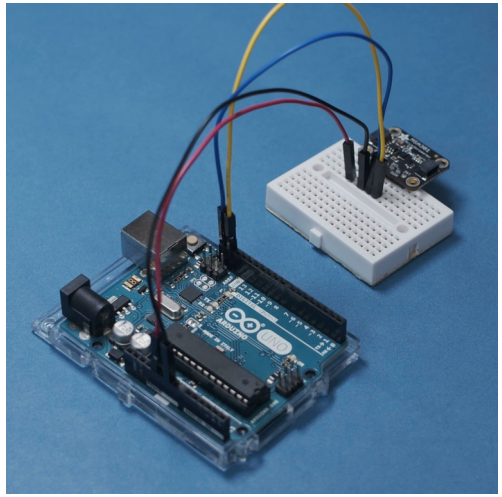
- LabVIEW :
  - Perhaps the most familiar
  - Graphical programming software for data flows (also control, automation)
  - Designed to support National Instruments hardware. Functions with 3rd party h/w as well
  - Runs on your PC / laptop
- Matlab/Simulink software
  - Similar, but without the direct h/w tie-in
  - Graphical or scriptable



For the Hobbiest

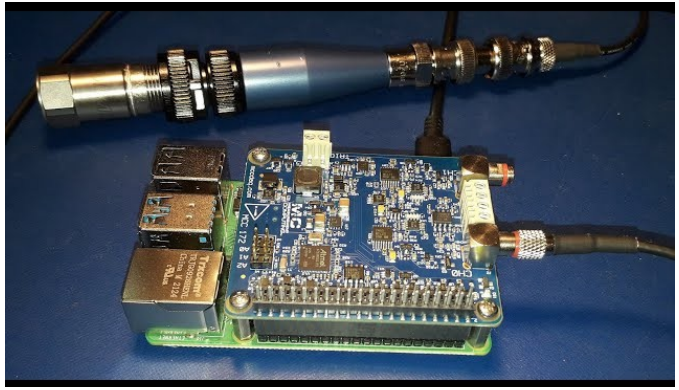
Arduino

- Single-board microcontroller
- Firmware programming via IDK, “C” or python
- Native analog GPIO, ADC/DAC, etc



Raspberry PI

- ARM based single-board computer
- Linux-based OS



Cheap and accessible! Get one!

Many sensor & DAQ extension boards available for both

- See : Adafruit, Sparkfun, etc

Not Just For the Hobbiest

CMS Internal Note

The content of this note is intended for CMS internal use and distribution only

22 September 2016 (v2, 17 February 2019)

Arduino for monitoring in the CMS experiment

F.Palmonari[1)(a)], E.Butz[(b)], A.Kaminskiy[(c)]
[(a)] *University of Wisconsin*
[(b)] *KIT, Karlsruhe*
[(c)] *Skobeltsyn Institute of Nuclear Physics, Lomonosov Moscow State University*

Abstract

During the long shutdown one of the Large Hadron Collider (LHC) we deployed a network of 562 temperature and humidity sensors inside and outside the solenoid volume of the Compact Muon Solenoid experiment (CMS). This network was used to validate the conditions for cold operation of the CMS silicon strips tracker detector. A low-cost monitoring system based on 13 Arduino Mega2560 boards with 1-wire standard readout was integrated in the CMS experimental cavern. The construction of this system involved the development of a custom Arduino-shield extension board and of a custom sensor that combines relative humidity and temperature sensing elements to output dew-point estimations. In this note we give a detailed description of the final system now used in daily operations of the CMS tracker.

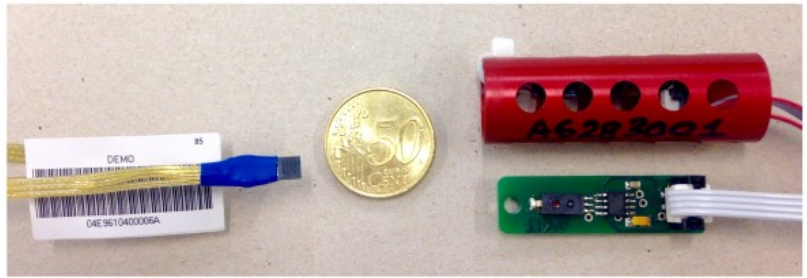


Figure 1: Sensors readout by the CMS Arduino system. On the left the digital thermometer DS18B20. On the right the custom made *multi-sensor* DS2438+HIH4030 with and without its anodized aluminium protection.

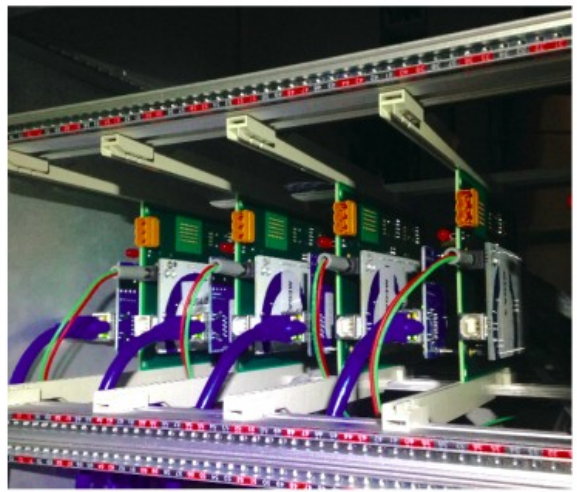


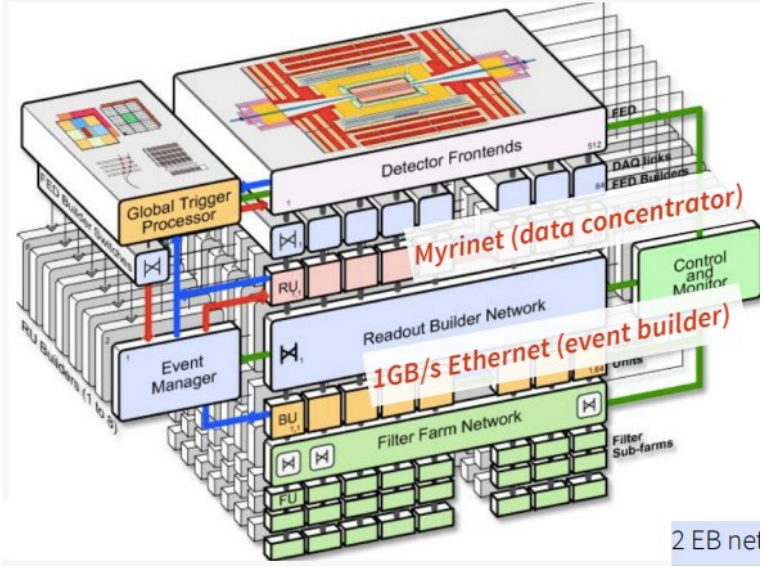
Figure 3: Near-side Arduino© crate in X2N37 located in the CMS experimental cavern.

HEP DAQ

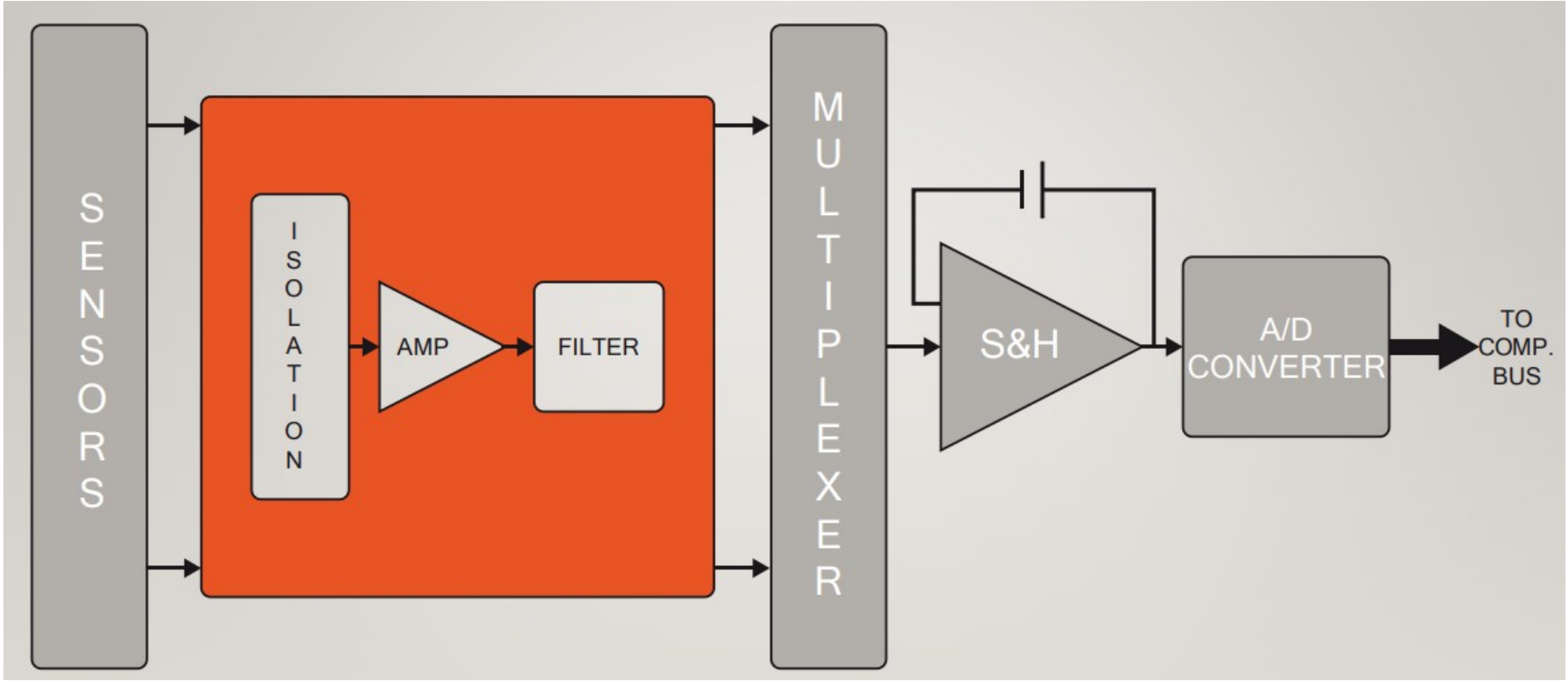
Complete commercial DAQ solutions generally not viable in HEP

- Custom sensors and electronics
- Environmental : remote, irradiated, cold
- Enormous channel counts
- Huge data rates and data volumes
- Often low latency requirements

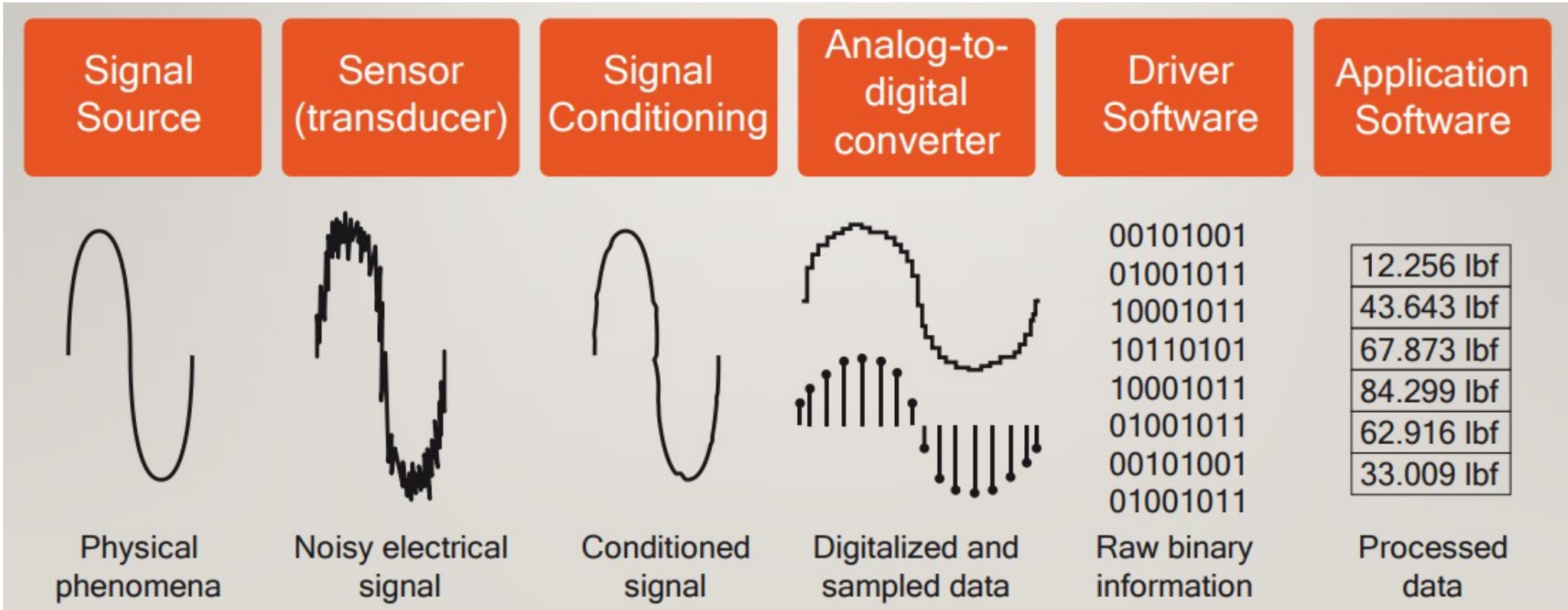
We have to design, implement, commission, and operate our own!



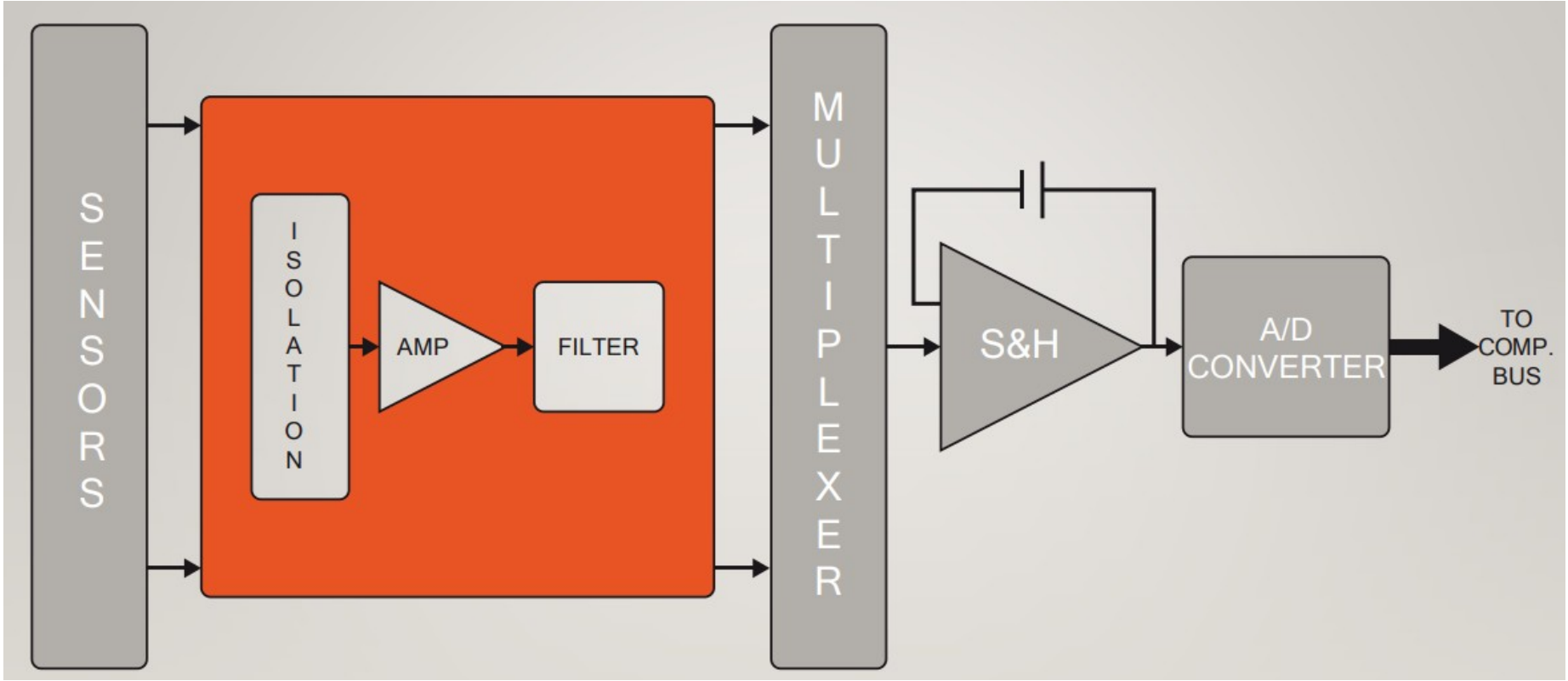
A Basic DAQ System



Basic DAQ Tasks



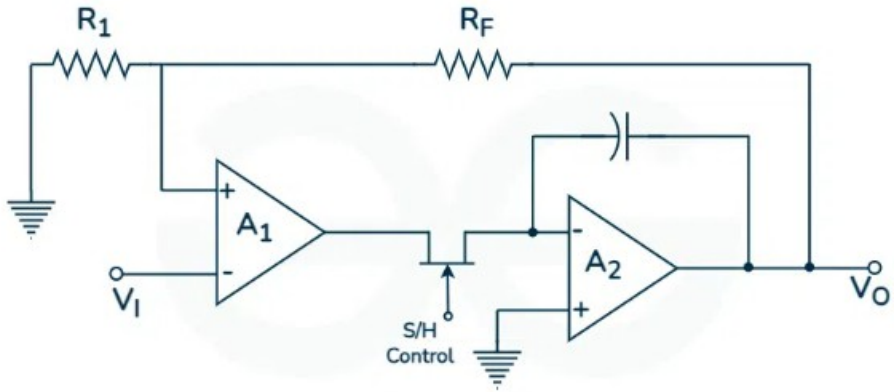
Something's Missing ...



Trigger Warning

A control signal to switch between sample and hold …

In general, need a signal to ”trigger” digitization, transmission, etc



- Sometimes this is simply a clock edge, ie: a periodic trigger
- But often more involved, eg:
  - A threshold discriminator
  - A coincidence trigger
  - Physics-based ...

In HEP, triggering is closely related to DAQ (aka “TDAQ”, more later)

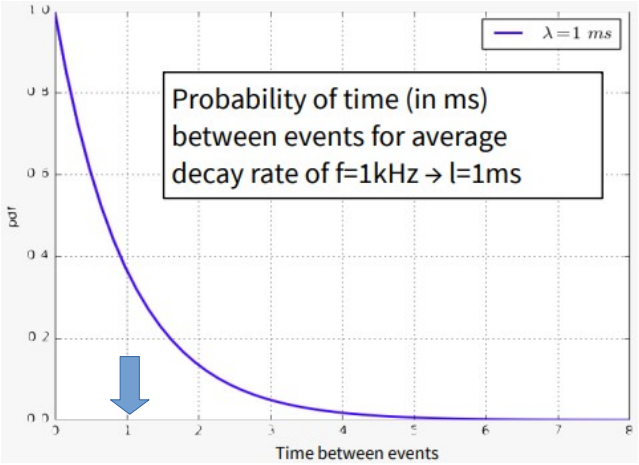
A Simple DAQ System

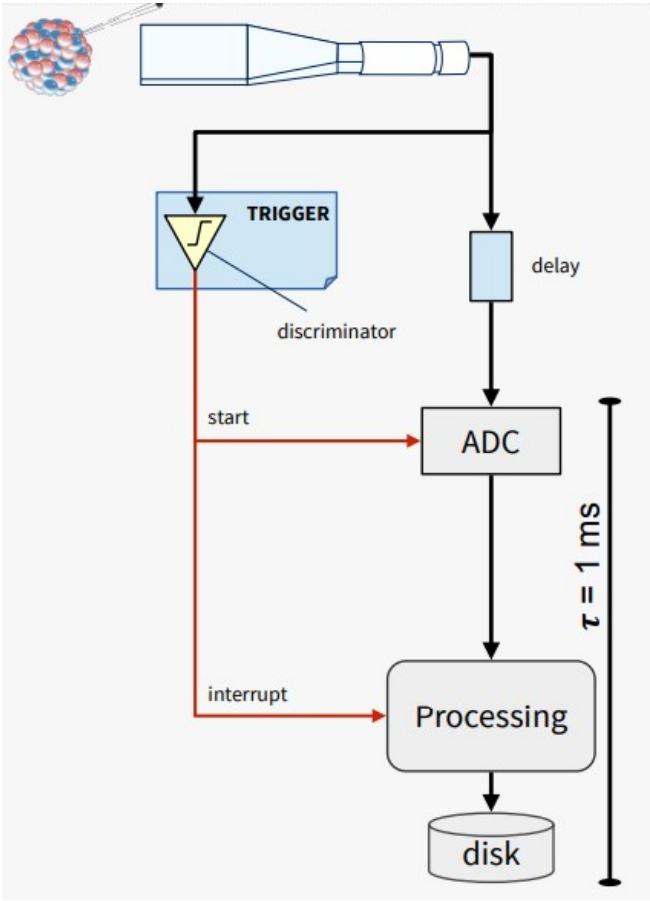
(borrowing liberally from A.Thea et al for STFC)

Physics Example

Beta decay

- Start with an ensemble of atoms
- Poisson probability for decay of individual atom within a time interval
- Time between decays is an exponential function
  - λ : mean time between decays
    - Consider 1ms
  - $f = \lambda^{-1}$: mean decay rate

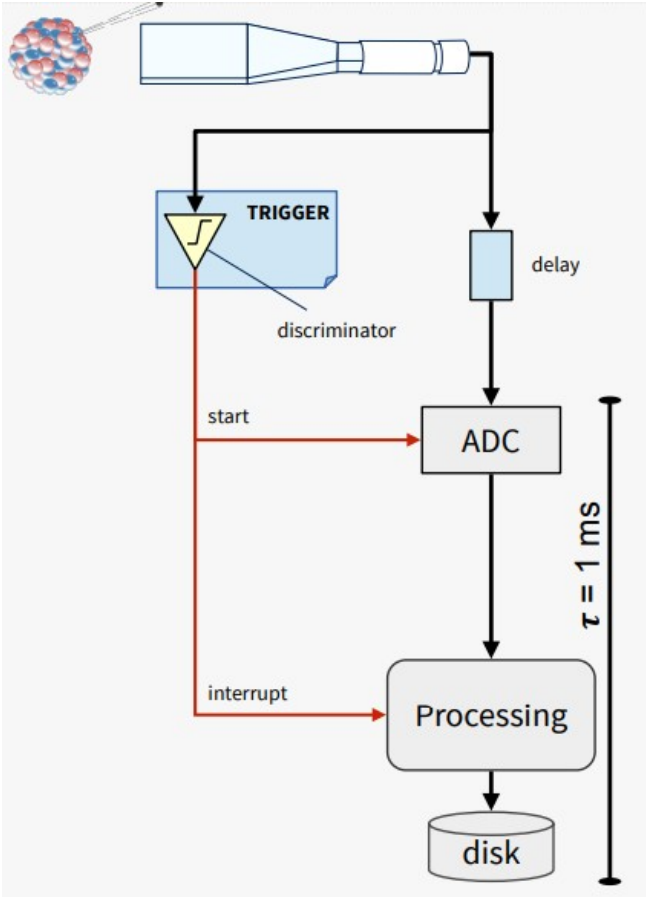




Physics Example

Beta decay

- $e^{\pm}$ generates an analog signal in a PMT
- That signal follows two paths
  - To an analog discriminator, compares with a set threshold, gives a digital output
  - To an ADC, with the input delayed to synchronize with the discriminator output
- The discriminator “triggers” the ADC digitization
- Once digitized data ready, data processing (eg: formatting, calibration corrected) triggered
- Processed data then stored on disk



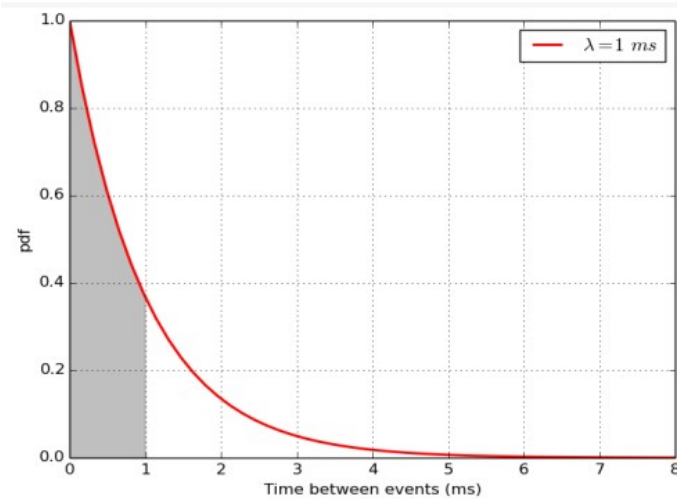
Physics Example

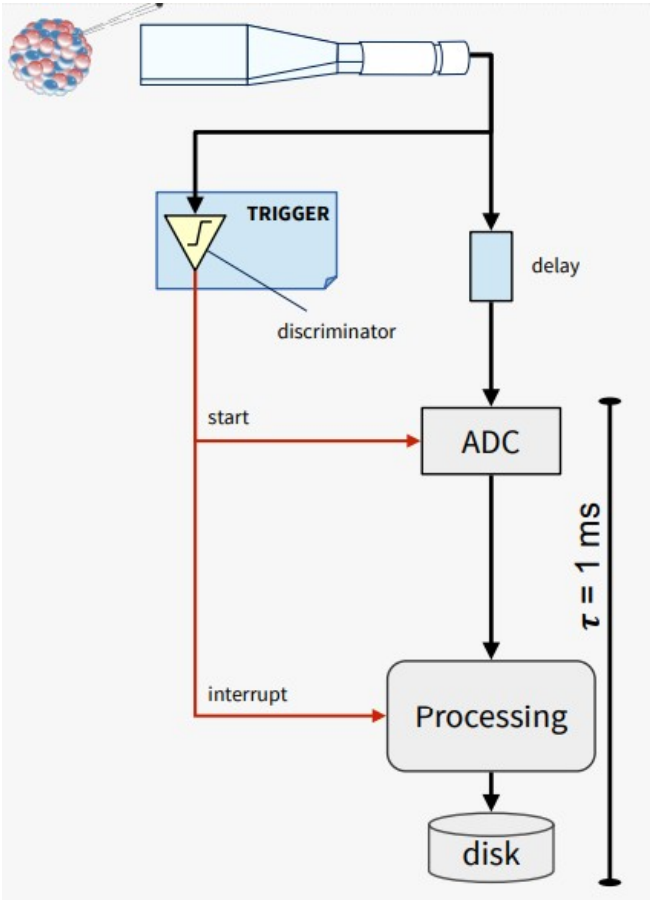
Digitization / processing chain takes some time to complete : latency

- Average decay time is λ = 1 ms, so design the chain to accommodate → τ = 1 ms

But time between decays is random

- What if rate fluctuates high?
- Inserting new data in the chain while processing risks corruption ...





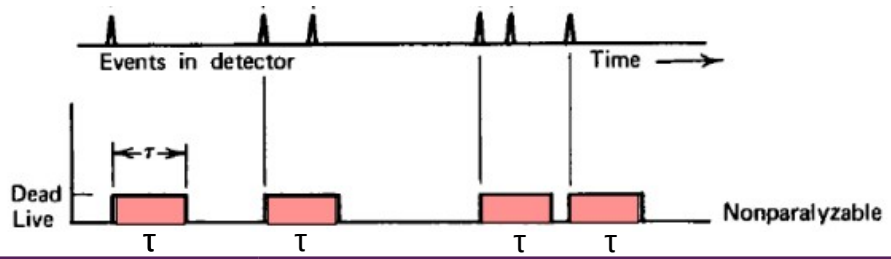
Physics Example

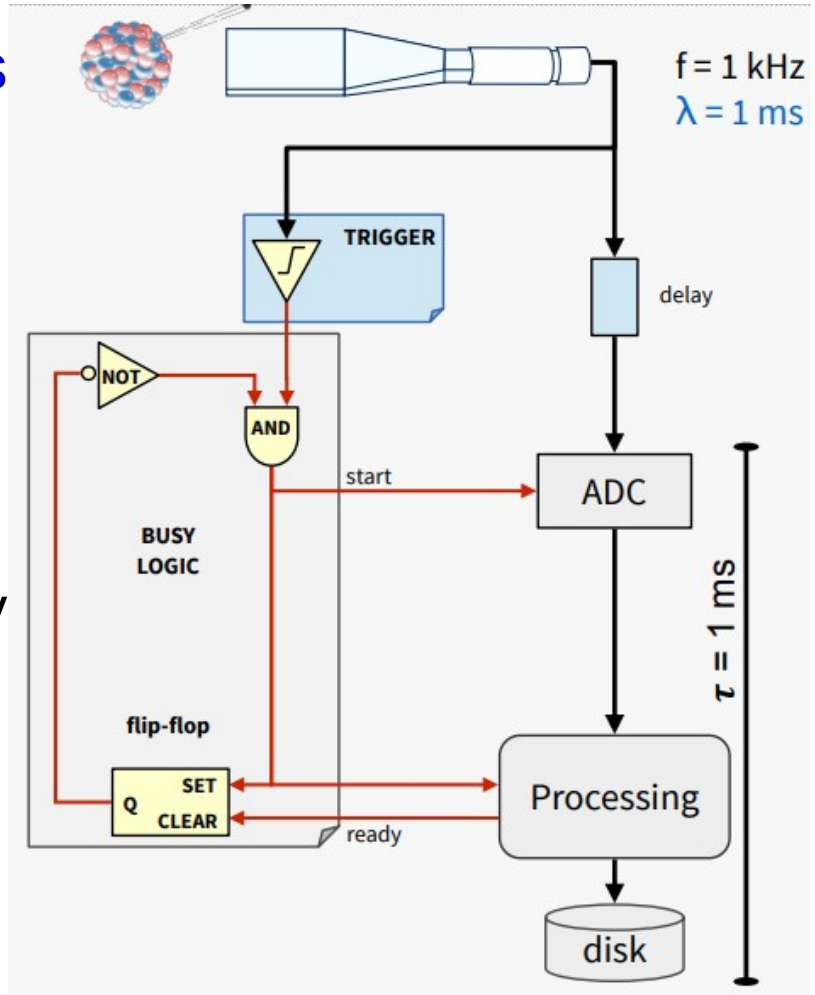
Protect processing by *inhibiting* new triggers when the chain is busy

- Limits max throughput to disk to τ^{-1} = 1 kHz
- This would be the case if, eg, we had a perfectly timed clock trigger …

The inhibit introduces *deadtime*

- Leading to a loss of data collection efficiency
- Ie : $f_{disk} < f_{physics}$

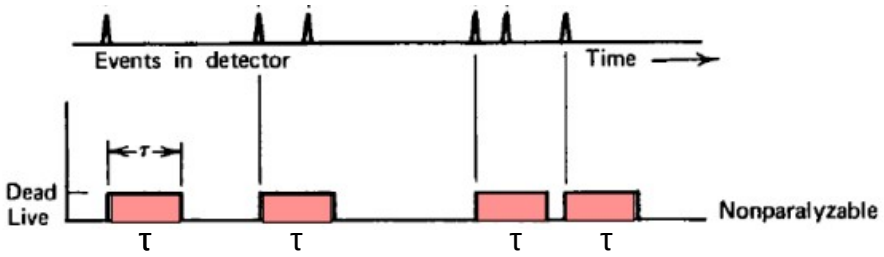




Physics Example

Determining f_{disk} :

- Referred to as r_{obs} below, and $f_{physics} = r$ ...



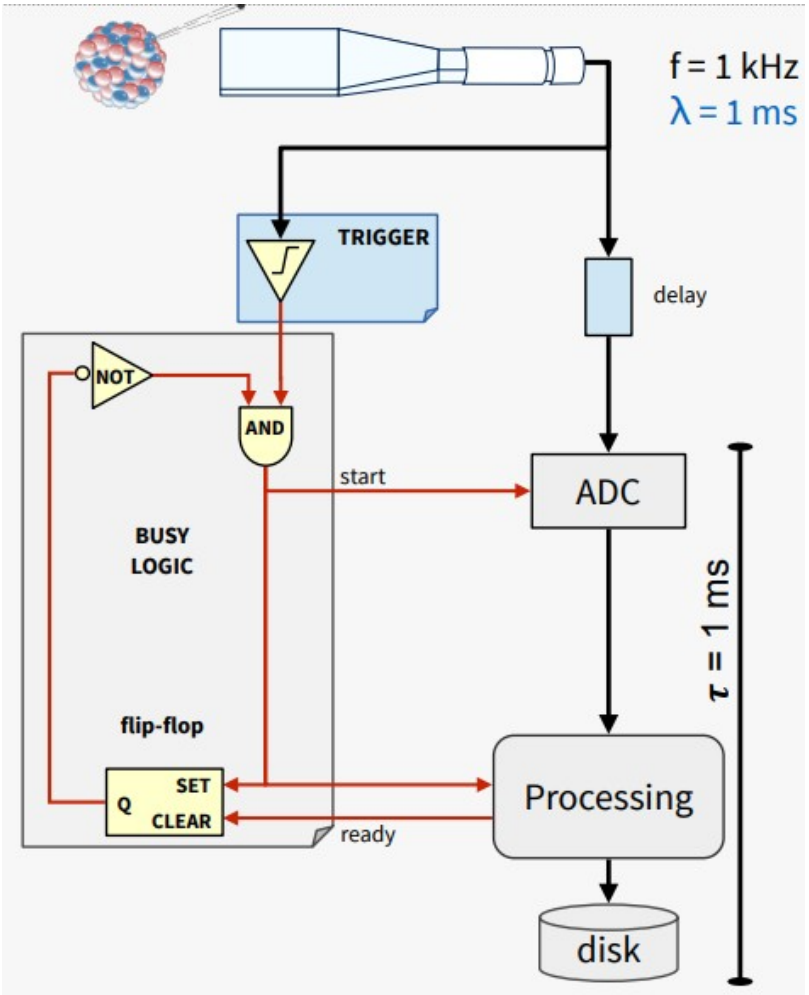
Loss rate = $r - r_{obs}$ or $n - m$ (in text)

Fraction dead = $r_{obs}\,\tau$ or $m\,\tau$

[Loss rate = $r\,(r_{obs}\,\tau)$ or $n\,(m\,\tau)$]

Fraction live = $(1 - r_{obs}\,\tau)$ or $(1 - m\,\tau)$

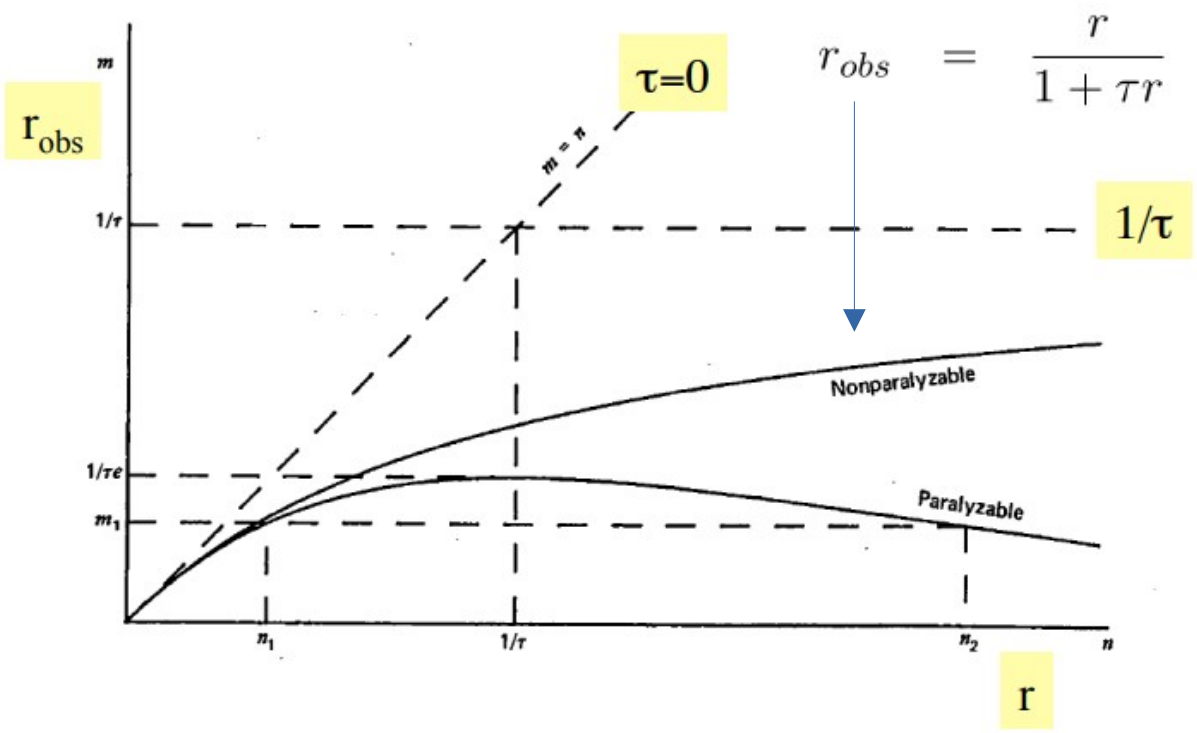
$r_{obs} = r\,(1 - r_{obs}\,\tau) \rightarrow r = r_{obs} / (1 - r_{obs}\,\tau)$

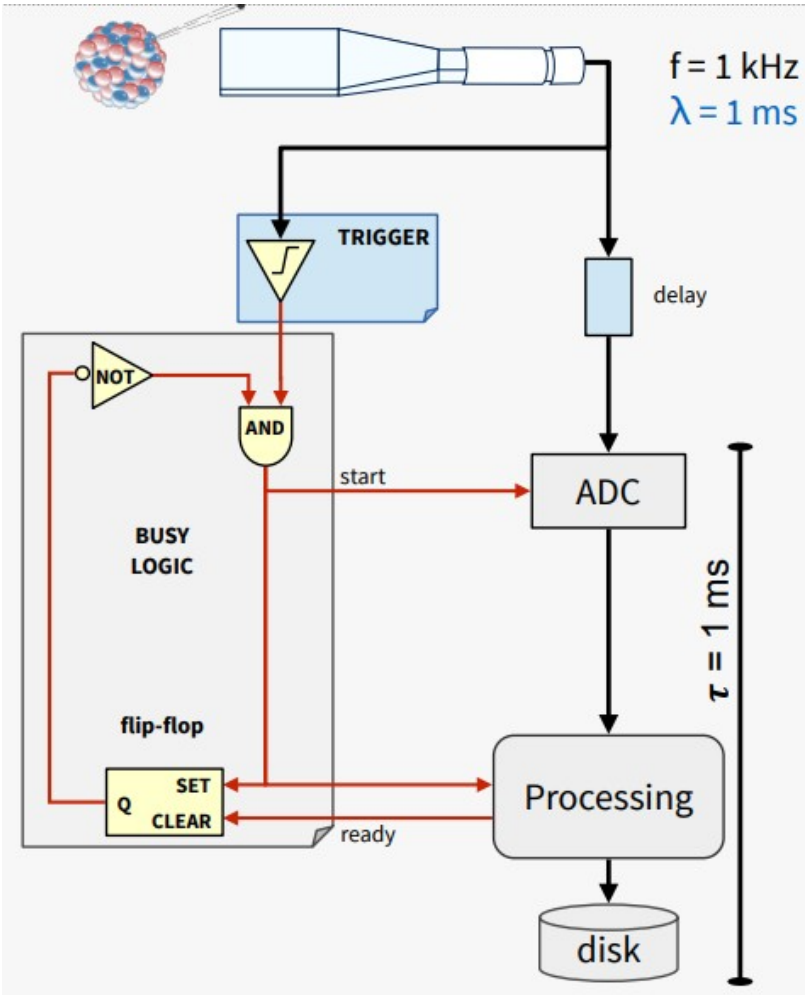


Physics Example

Determining f_{disk} :

- Referred to as r_{obs} below ...





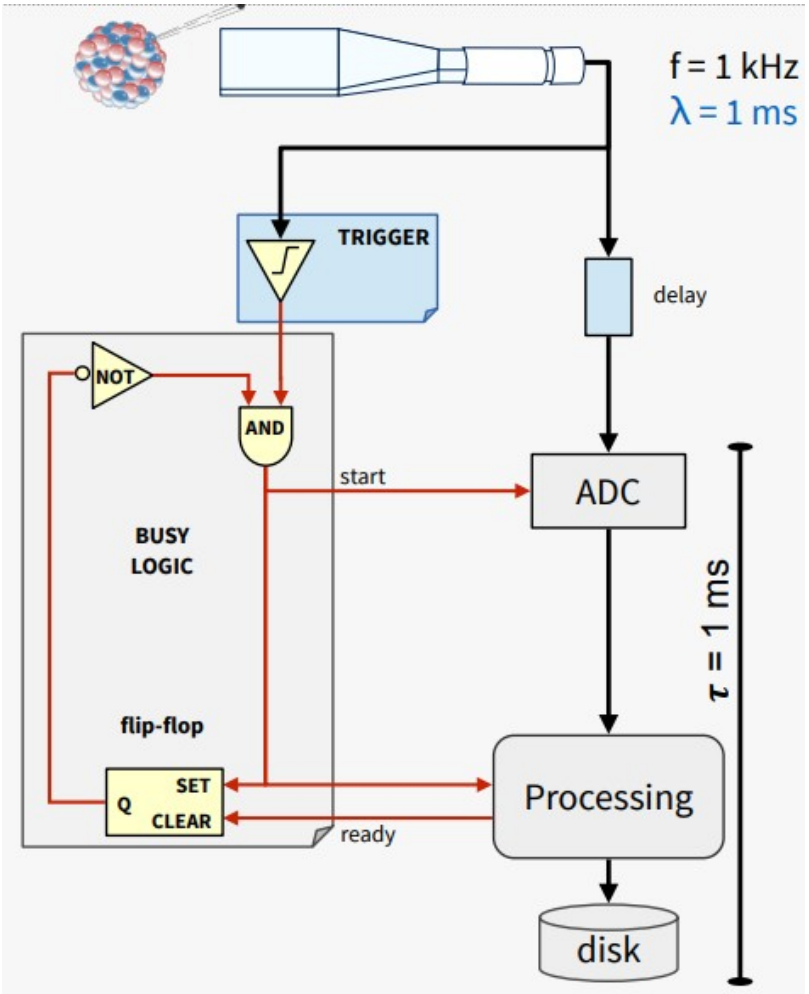
Physics Example

Output rate lower than input rate :

$$r_{obs} = \frac{r}{1+\tau r} < r$$

Therefore DAQ inefficiency :

$$\epsilon = \frac{r_{obs}}{r} = \frac{1}{1+\tau r} < 1$$

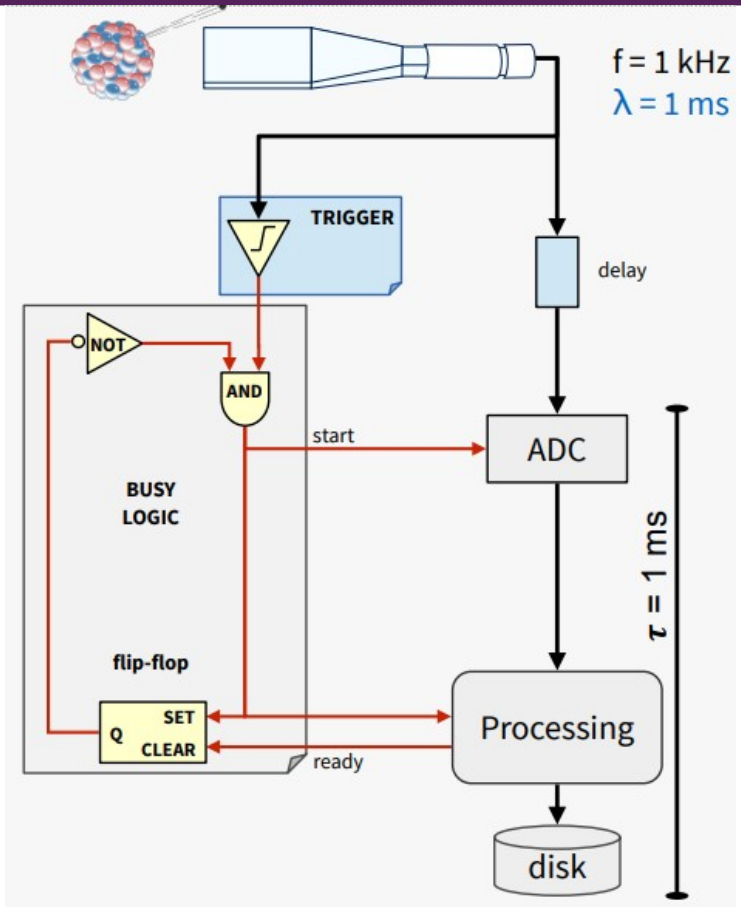


Physics Example

What is the efficiency in our example?

$$r_{obs} = \frac{1\ \text{kHz}}{1 + (1\ \text{ms})(1\ \text{kHz})} = 0.5\ \text{kHz}$$

$$\epsilon = 50\%$$



To improve efficiency, can try to reduce τ, eg:

$$\epsilon = 0.99 \quad \rightarrow \quad \tau = 0.01\ \text{ms}$$

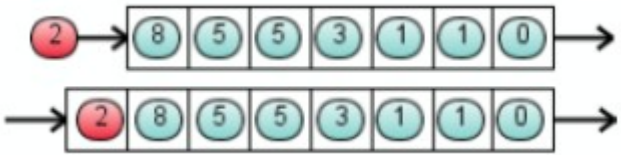
$$\epsilon = \frac{r_{obs}}{r} = \frac{1}{1 + \tau r}$$

- For instance, via parallelization : *more pipelines*
- Here would need a factor of 100 over capacity beyond nominal … costly!

Physics Example

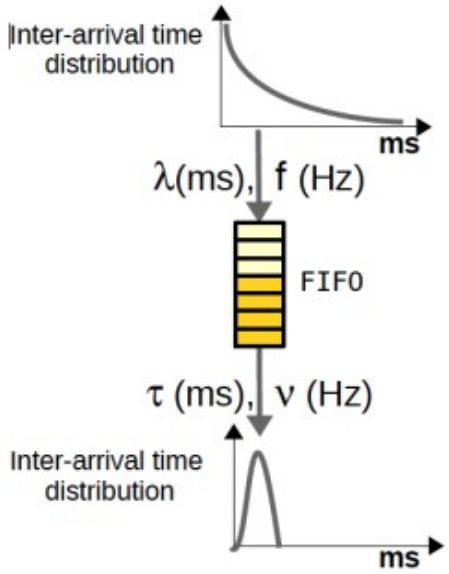
Another approach : *derandomization*

- Buffer the data with a FIFO until processing chain is available
  - FIFO = First In, First Out queue
  - Depth given by number of cells
  - Both h/w and s/w implementations



Absorbs input rate fluctuations

- Smooths the distribution of arrival time of data at the input to the ADC
- Processing start becomes more periodic
- But adds to the overall latency



Physics Example

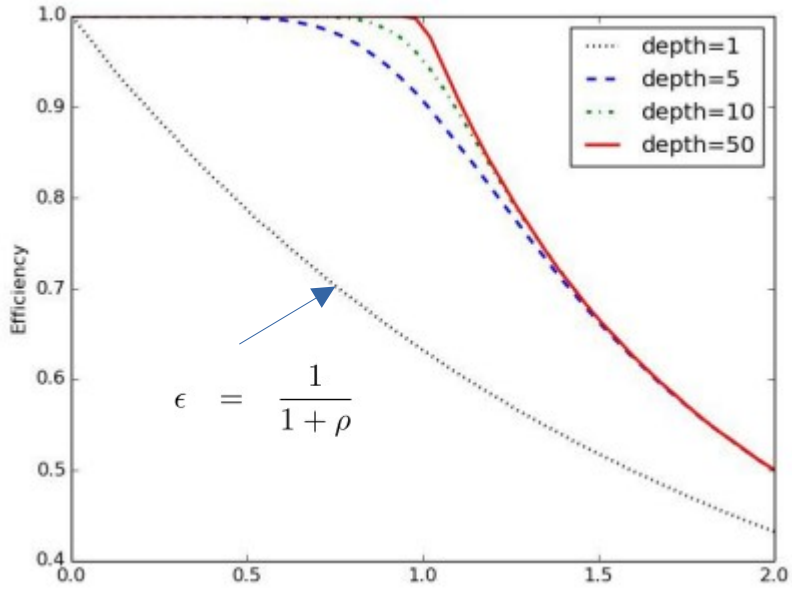
Result : efficiency now dictated by *buffer occupancy*

And occupancy depends on nominal rate and buffer depth

Study efficiency as a function of traffic intensity : ρ = τr

- ρ > 1 : system overloaded
- ρ << 1 : system over-designed
- ρ ~ 1 : system can achieve high efficiency with small buffer depth

Influence of buffer depth assessed using simulation in most cases

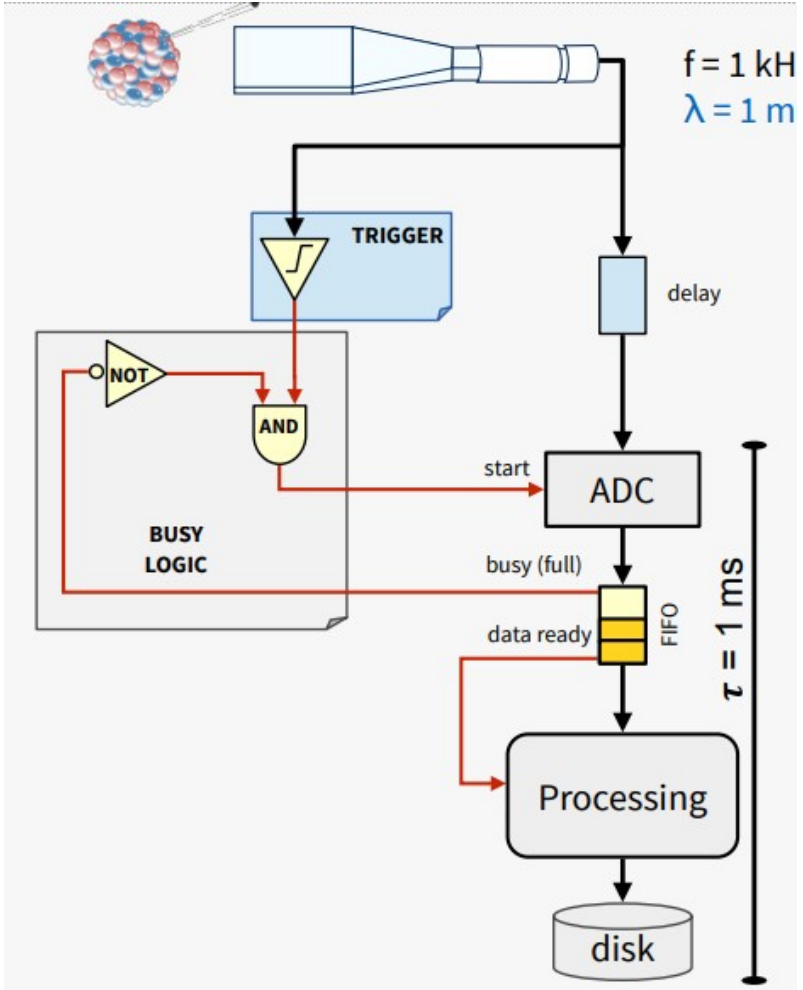


Physics Example

Processing now retrieves data from the queue when ready and data is available

Busy logic must monitor occupancy

- Otherwise FIFO can overflow
- Assert trigger inhibit when FIFO close to full



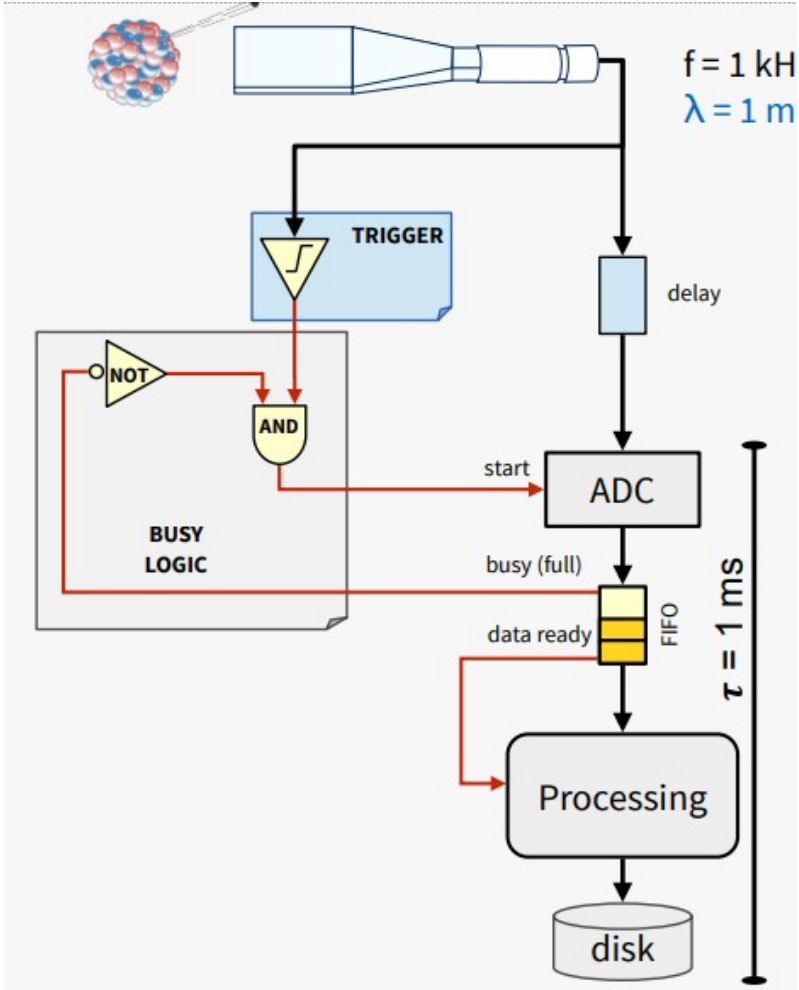
Physics Example

Processing now retrieves data from the queue when ready and data is available

Busy logic must monitor occupancy

- Otherwise FIFO can overflow
- Assert trigger inhibit when FIFO close to full

The FIFO adds latency, therefore delay



Physics Example

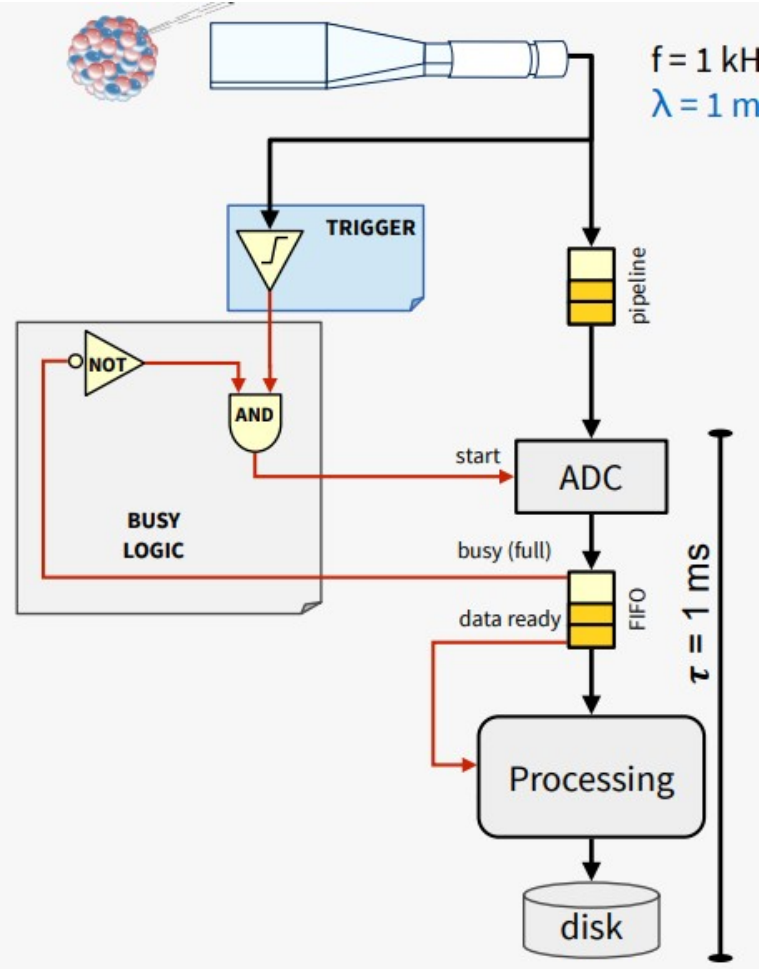
Processing now retrieves data from the queue when ready and data is available

Busy logic must monitor occupancy

- Otherwise FIFO can overflow
- Assert trigger inhibit when FIFO close to full

The FIFO adds latency, therefore delay

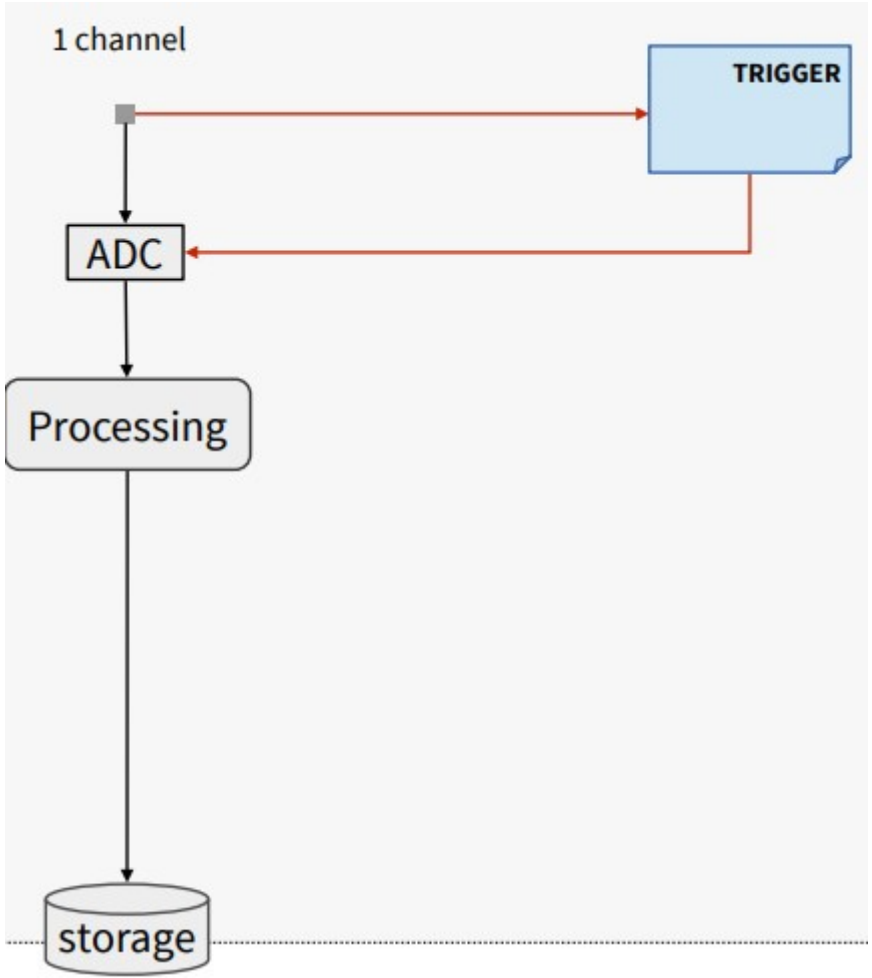
- Can replace the ADC delay with an analog equivalent (“pipeline”)
- Heavily used at the LHC : *frontend buffers*



DAQ Scaling

(borrowing liberally from A.Thea et al for STFC)

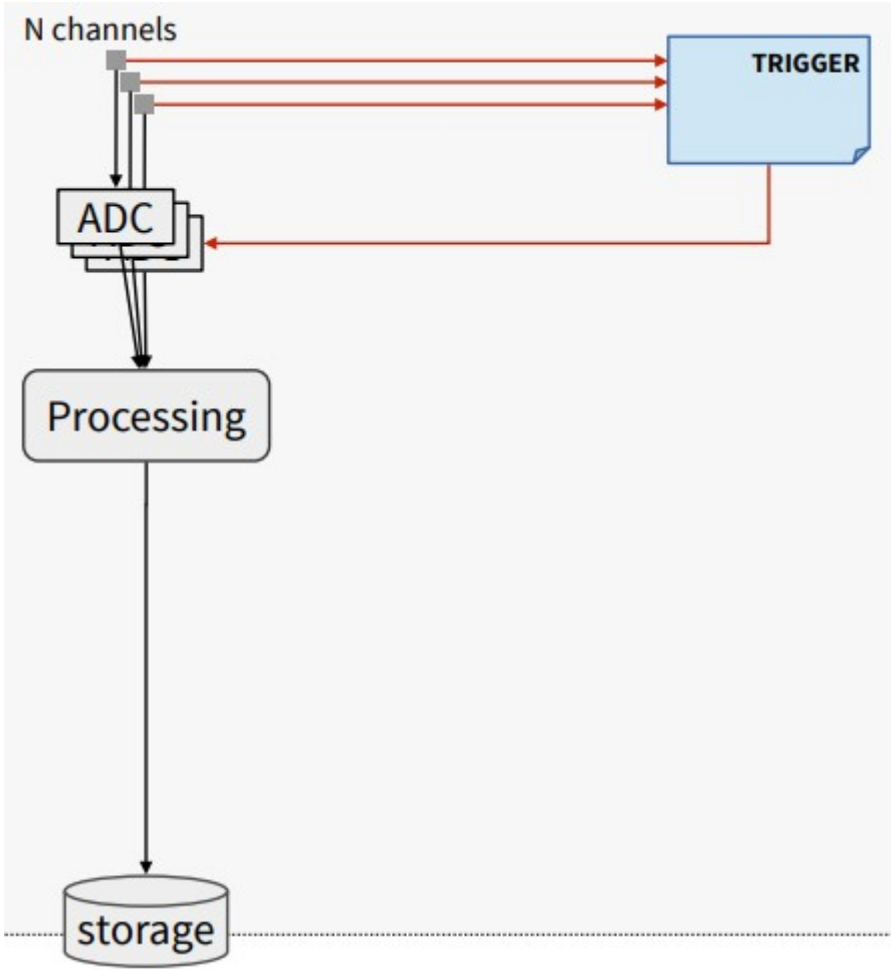
Scaling Up



Frontend data conditioning

- Digitization (often), buffering

Scaling Up



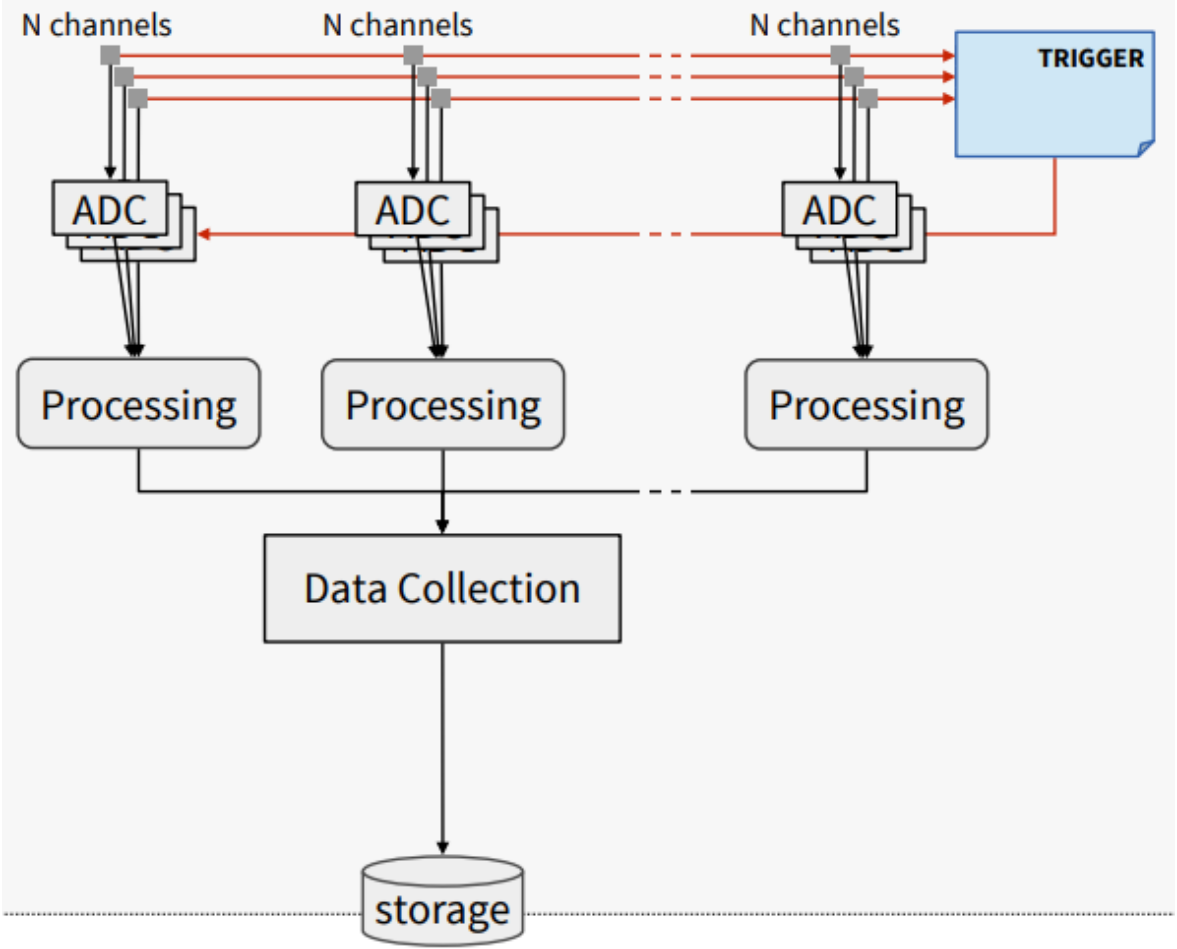
Frontend data conditioning

- Digitization (often), buffering

Frontend readout

- Sampling, formatting, buffering

Scaling Up



Frontend data conditioning

- Digitization (often), buffering

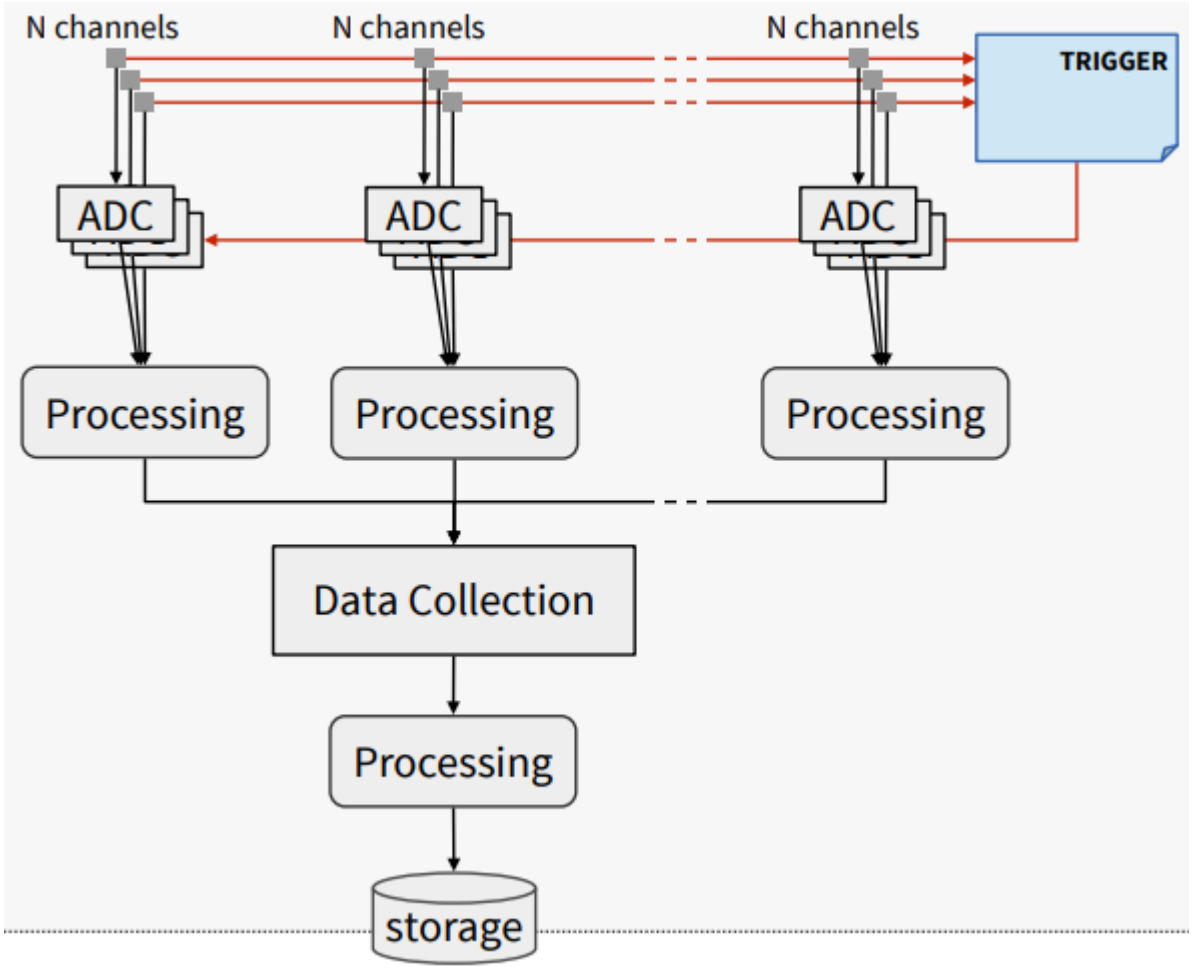
Frontend readout

- Sampling, formatting, buffering

Event building

- Data assembly, buffering

Scaling Up



Frontend data conditioning

- Digitization (often), buffering

Frontend readout

- Sampling, formatting, buffering

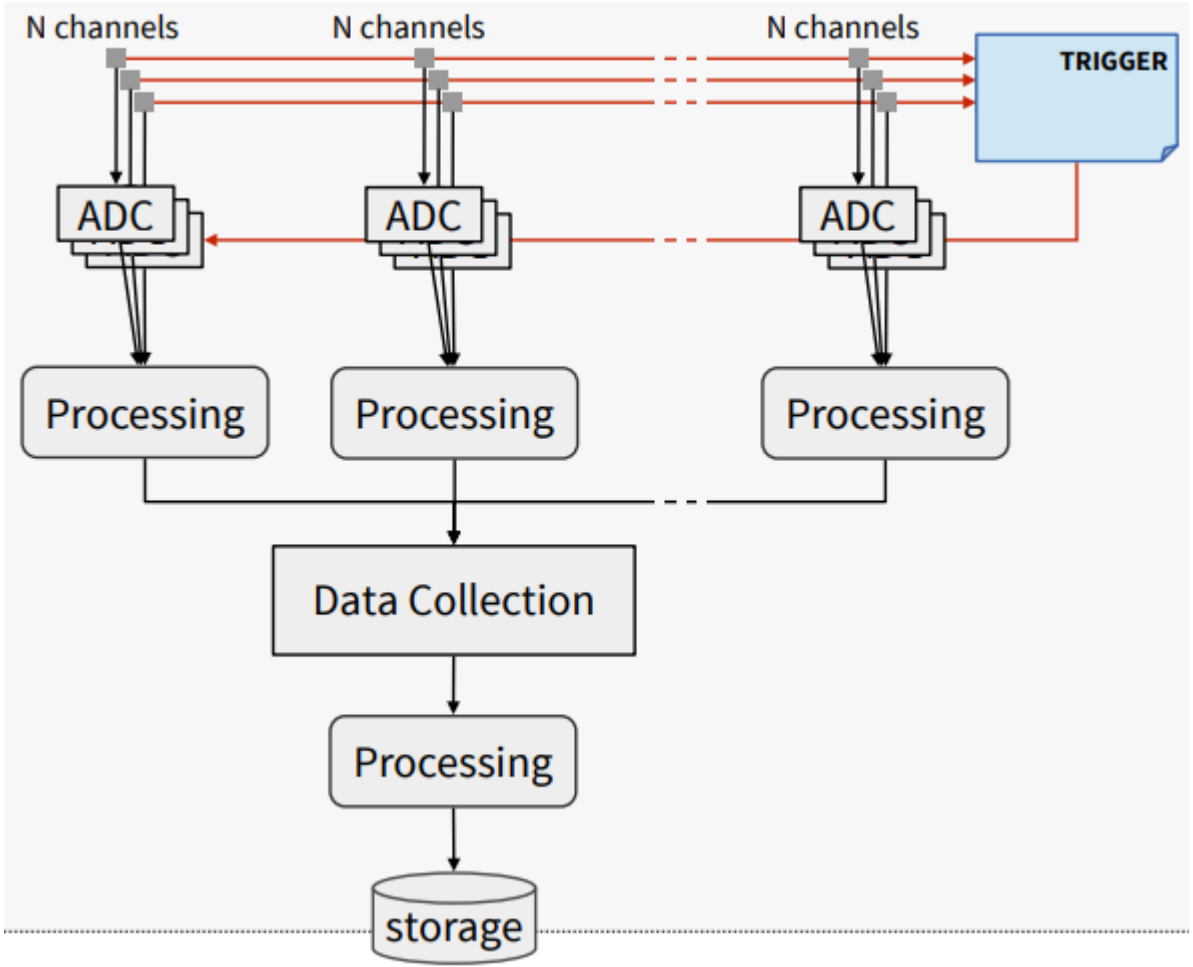
Event building

- Data assembly, buffering

Event filtering

- Select/reject, event buffering

Scaling Up



Frontend data conditioning

- Digitization (often), buffering

Frontend readout

- Sampling, formatting, buffering

Event building

- Data assembly, buffering

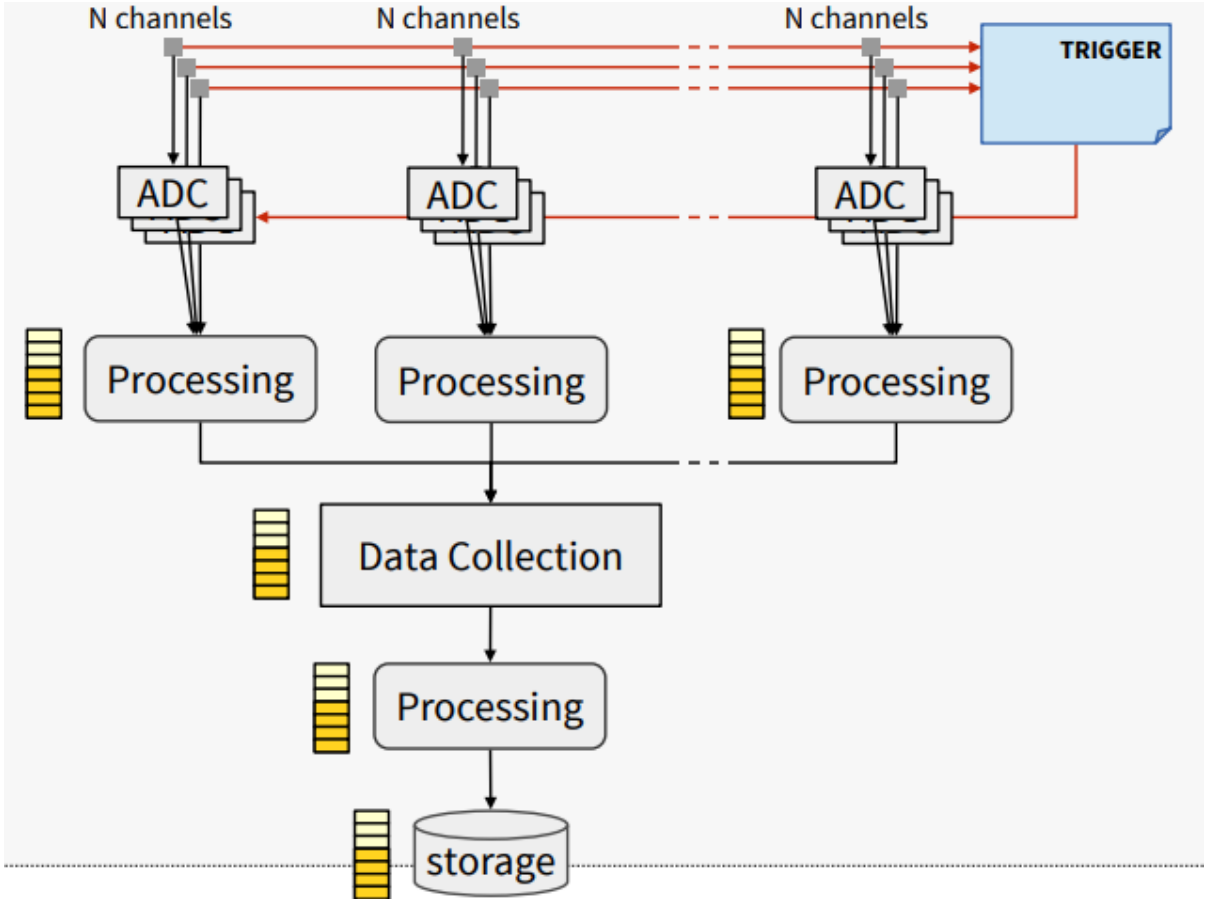
Event filtering

- Select/reject, event buffering

Event storage

- File logging, file buffering

Scaling Up

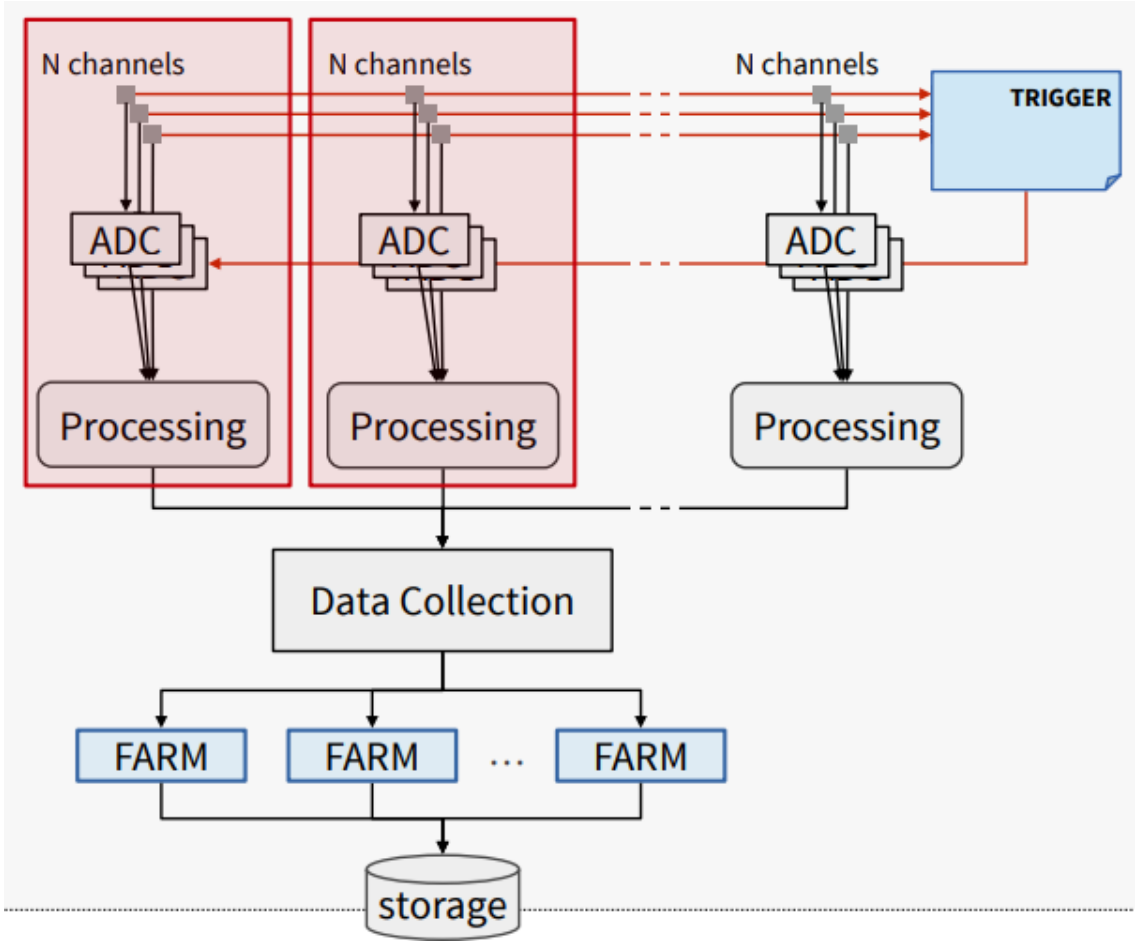


Buffering needed at all stages

No buffers should overflow

- Occupancy at one stage used to inhibit the preceding
- Inhibit propagates upstream until it reaches the trigger
- Referred to as “back-pressure”

Implementation



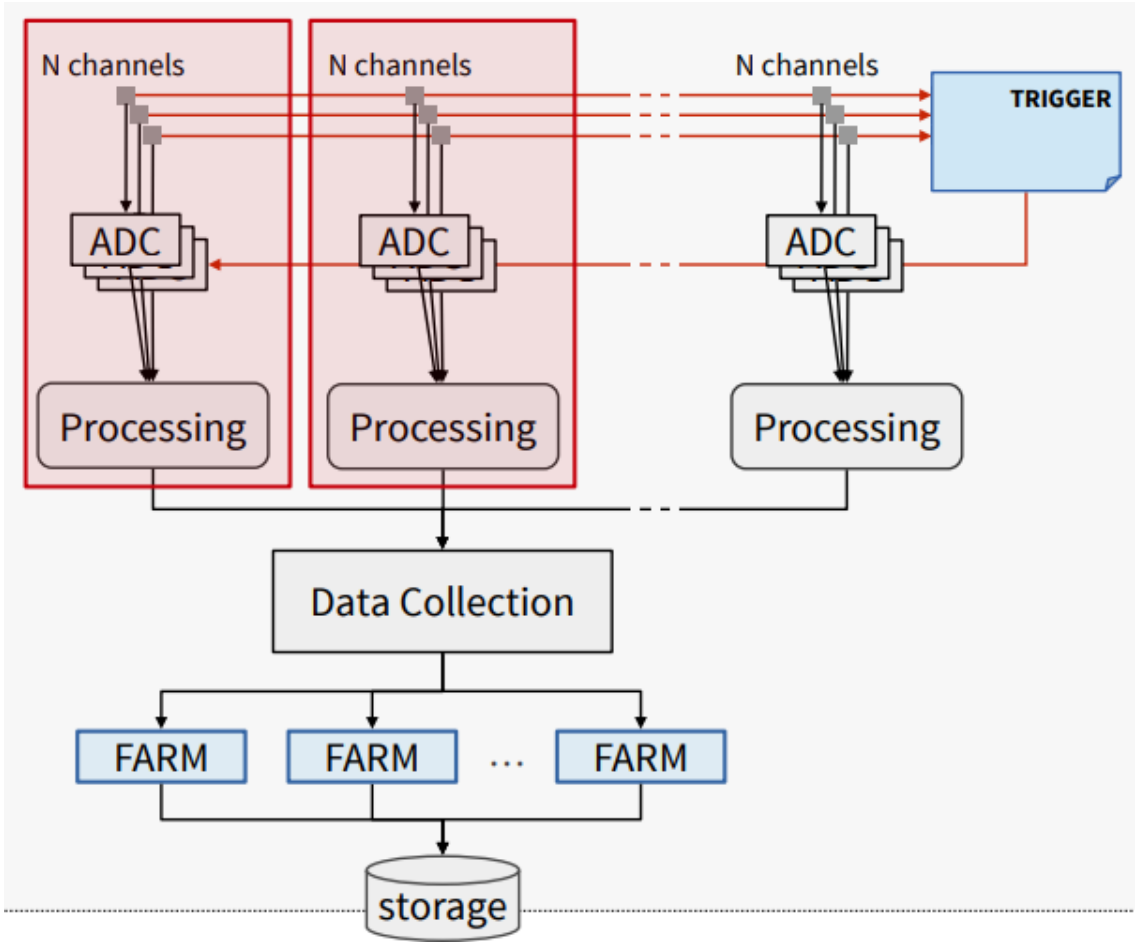
A Multi-tiered DAQ

- With basic functional units replicated within each tier

Ingredients

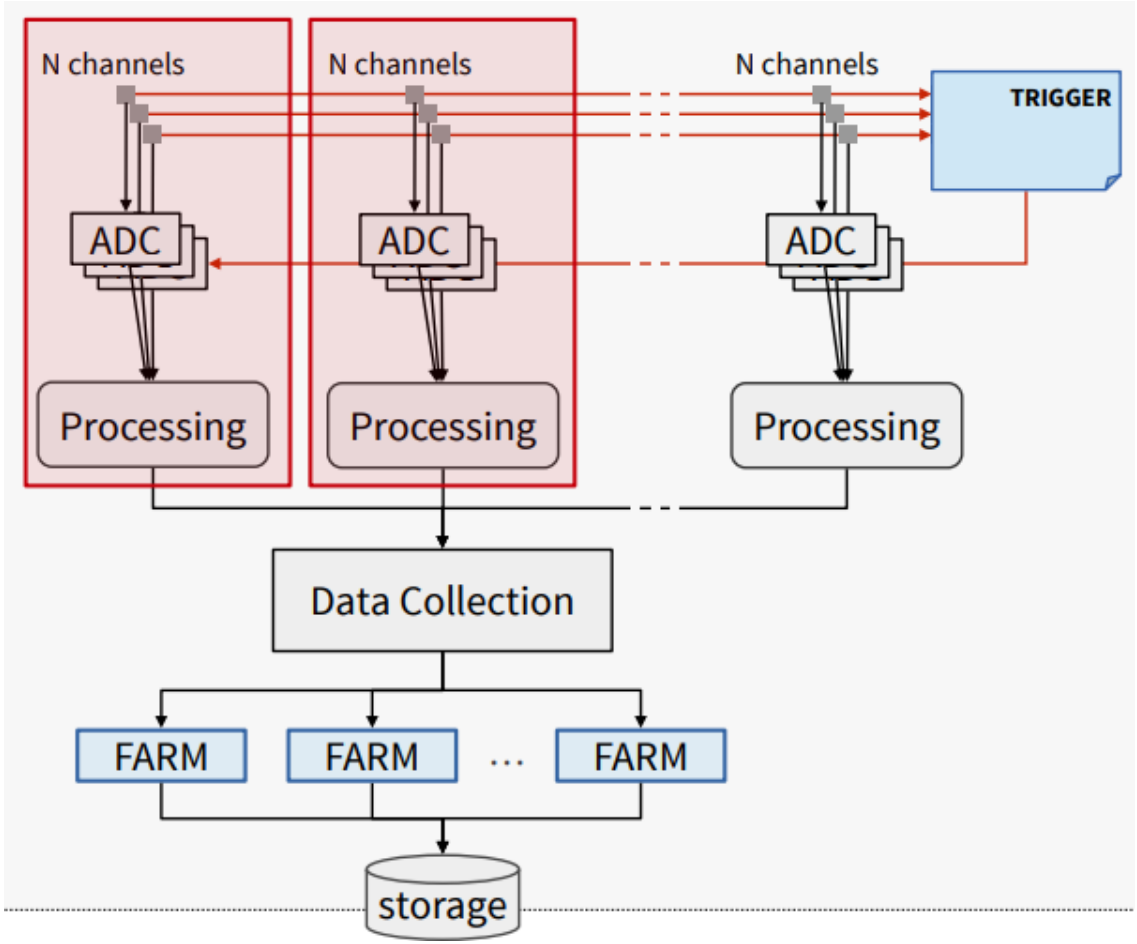
- Detector structures
- Readout & control cards
- Backend crates
- Racks
- Server farm

Implementation



Different technologies employed at different levels

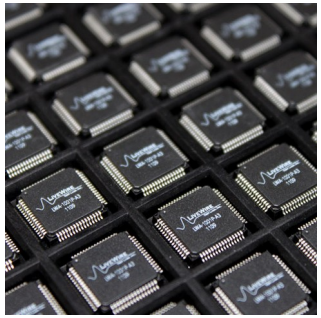
Implementation



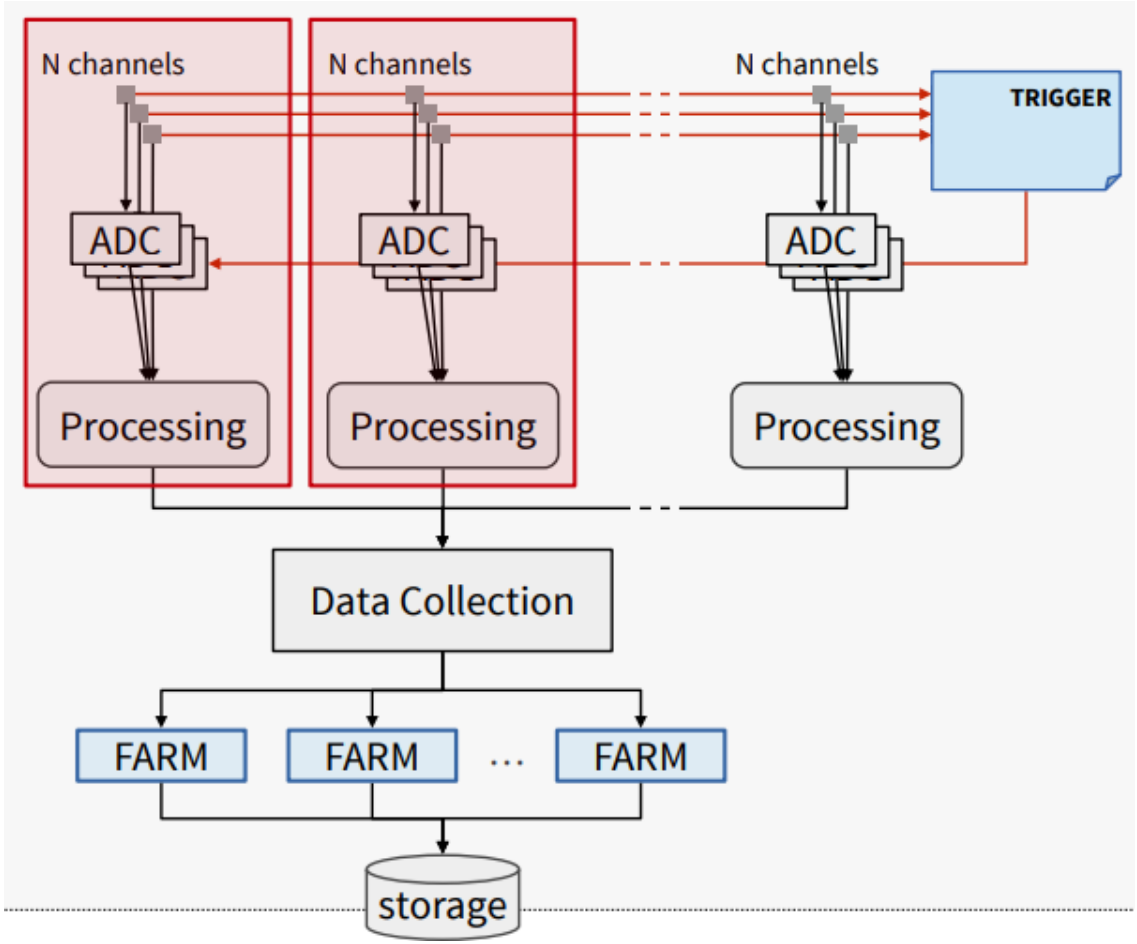
Different technologies employed at different levels

Frontend : ASICs

- Custom circuits
- Often radiation tolerance
- High speed, deterministic
- Buffering is expensive ...



Implementation

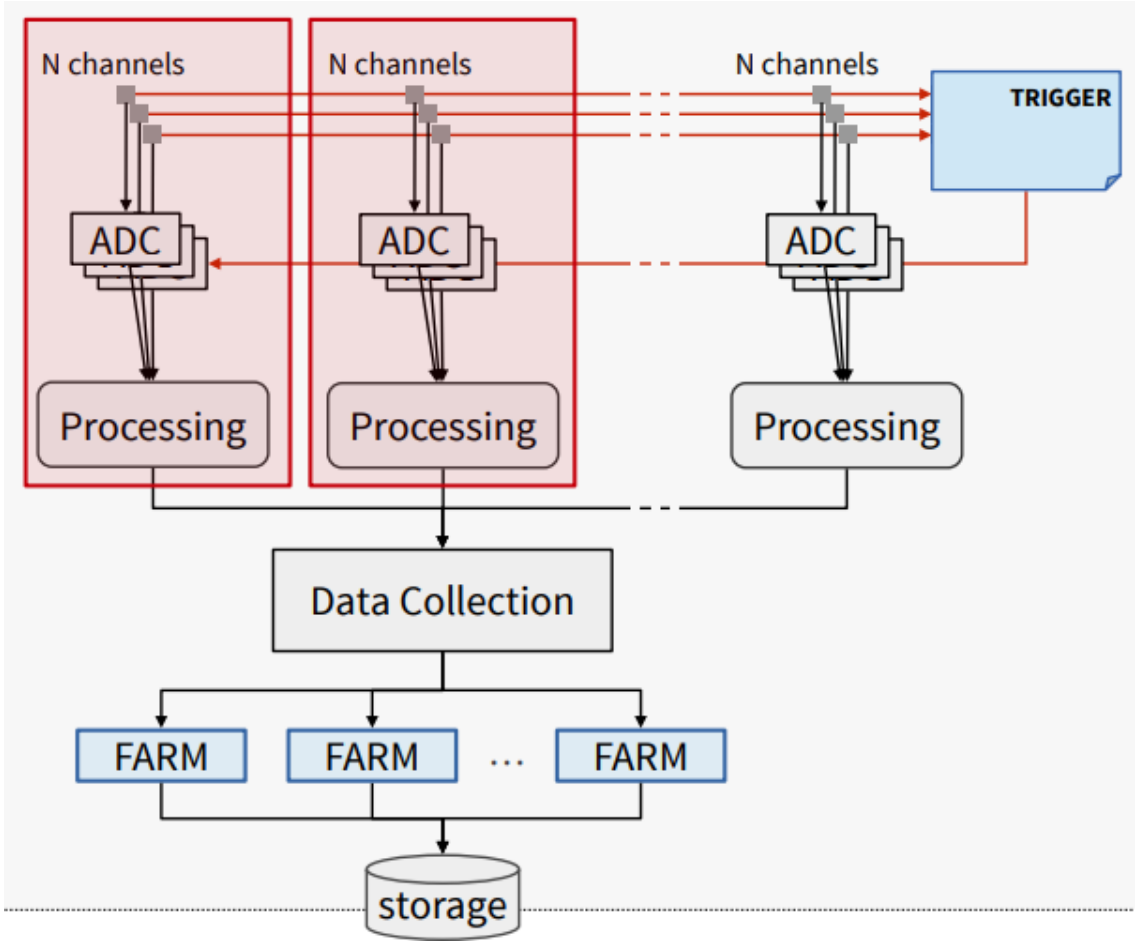


Different technologies employed at different levels

Backend processing : FPGAs

- Customizable logic
- Real time performance
- Resources : RAM, DSPs, MGTs, etc

Implementation



Different technologies employed at different levels

Event handling : CPUs, GPUs

- Relaxed latency requirements allows for batch processing
- Resources available for complex algorithms

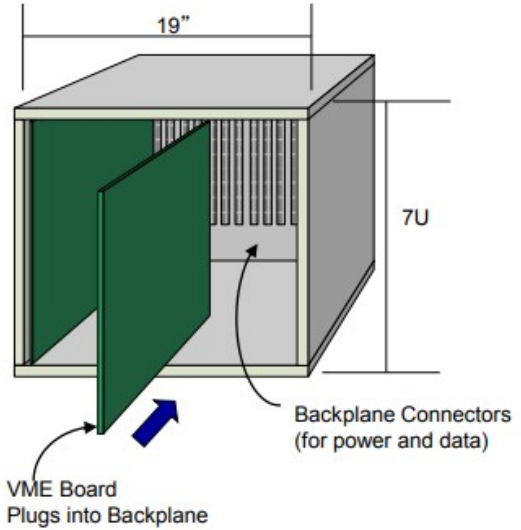
Implementation

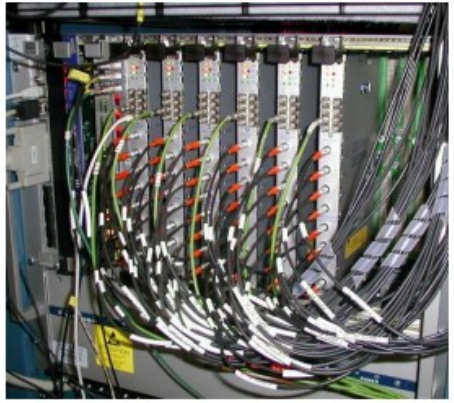
Backend FPGAs hosted on custom PCBs

- Providing upstream and down stream connectivity
- Interfaces to trigger and timing
- User access via microcontroller, sytem on chip, system on module, etc …

Backend boards housed in crate / shelf :

- Providing power, cooling, networking, etc
- Often extending in the rear with additional boards
- Boards generally communicate via the backplane





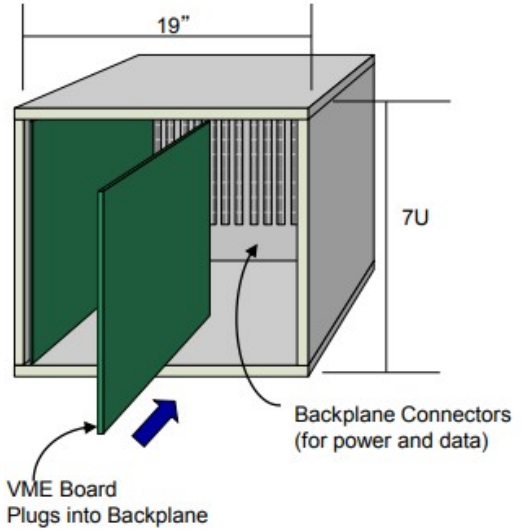
Implementation

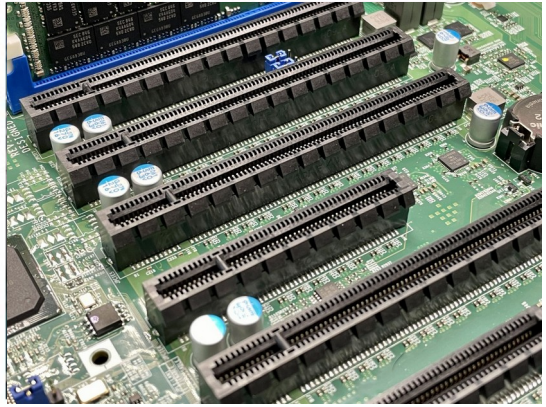
Backend FPGAs hosted on custom PCBs

- Providing upstream and down stream connectivity
- Interfaces to trigger and timing
- User access via microcontroller, sytem on chip, system on module, etc …

Backend boards housed in crate / shelf :

- Providing power, cooling, networking, etc
- Often extending in the rear with additional boards
- Boards generally communicate via the backplane
  - Similar to peripheral cards in a PC

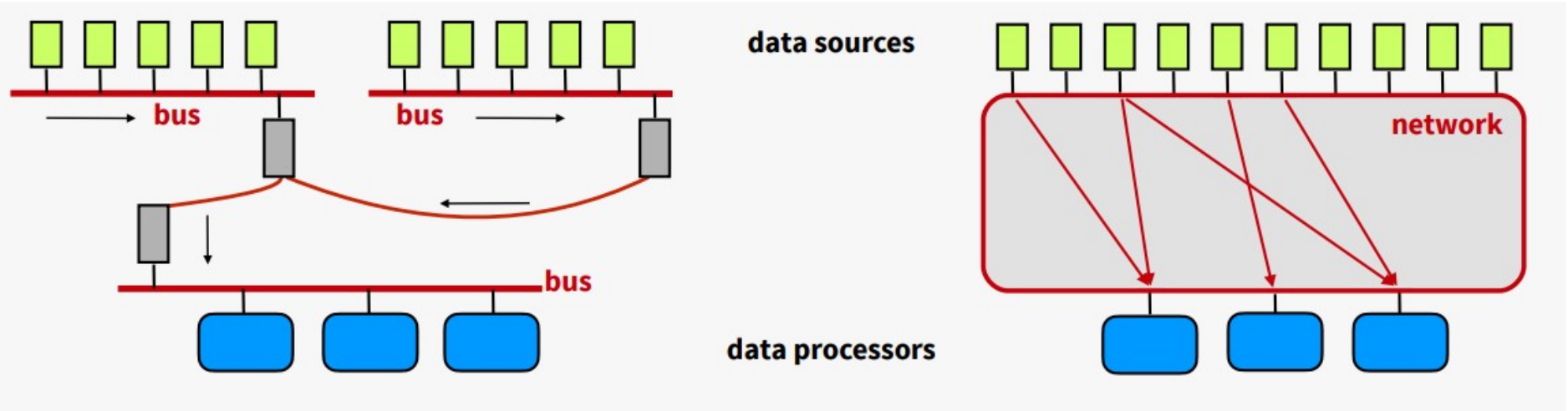




Implementation

Communication schemes : bus vs network

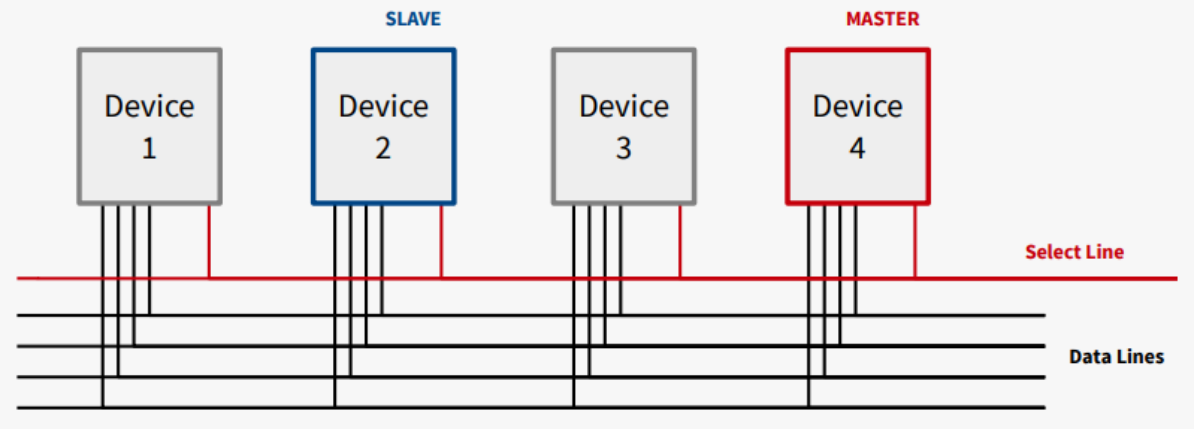
- Both have their advantages



Implementation

Busses : communication over shared electrical lines

- Access to the comm channel must be arbitrated
- Master/slave transactions
- Unique bus IDs / addresses
- Examples : PCIe, USB, NIM, VME



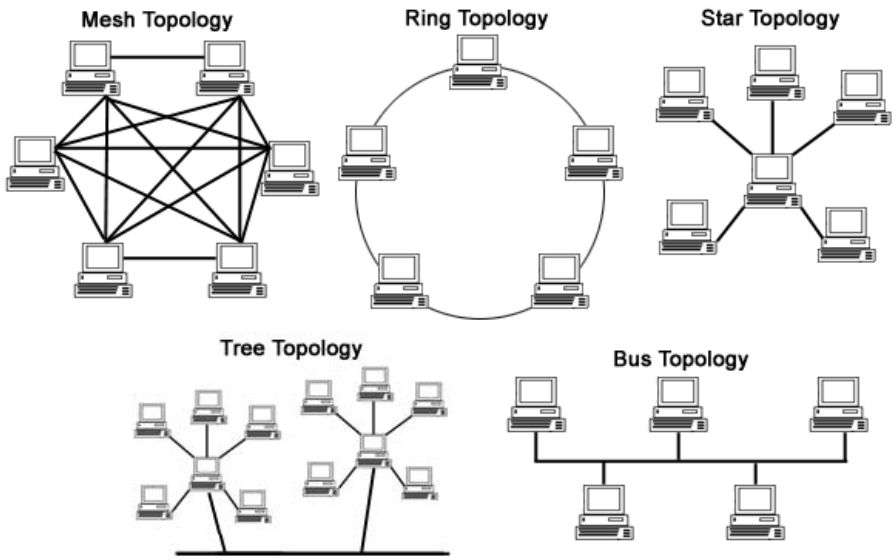
Considerations

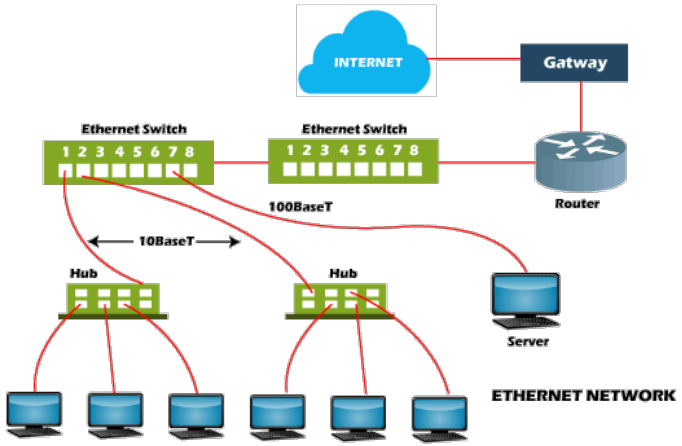
- Easy to add a new device
- Number of devices limited
- Typically slow
  - Bandwidth limited by bus width
  - Frequency by physical length

Implementation

Network : point-to-point communication among peers

- Electrical, optical, wireless
- Switched networks, different topologies
- Unique network addresses
- Examples : Ethernet, Infiniband

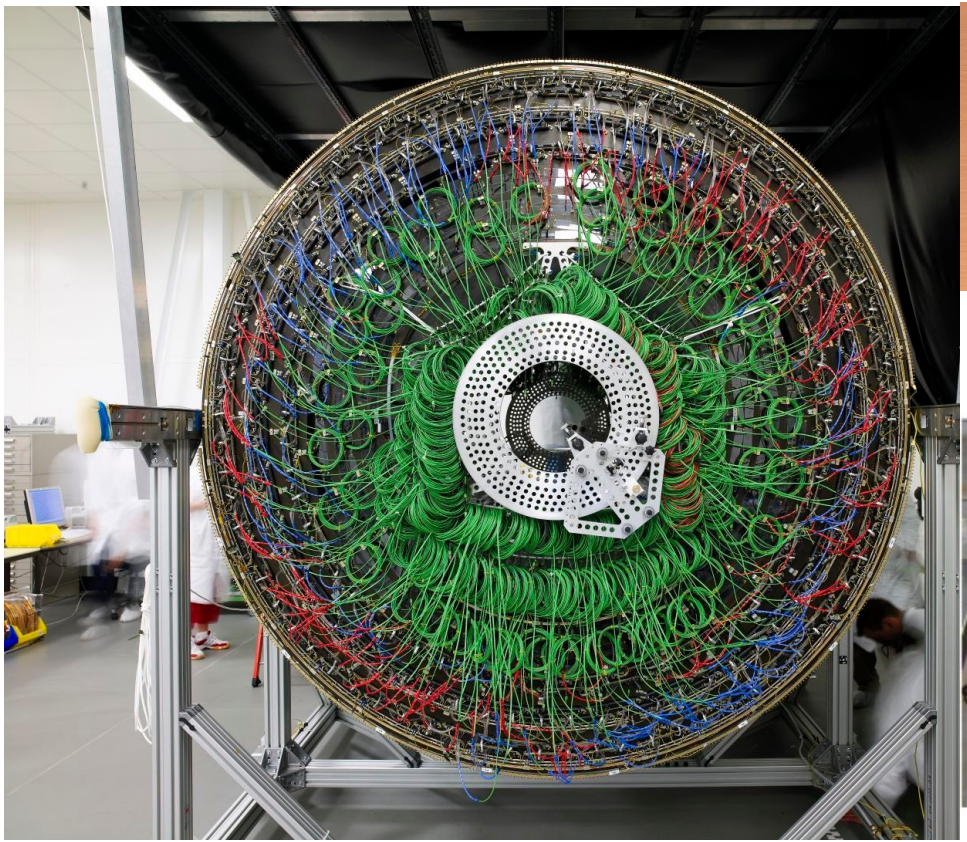


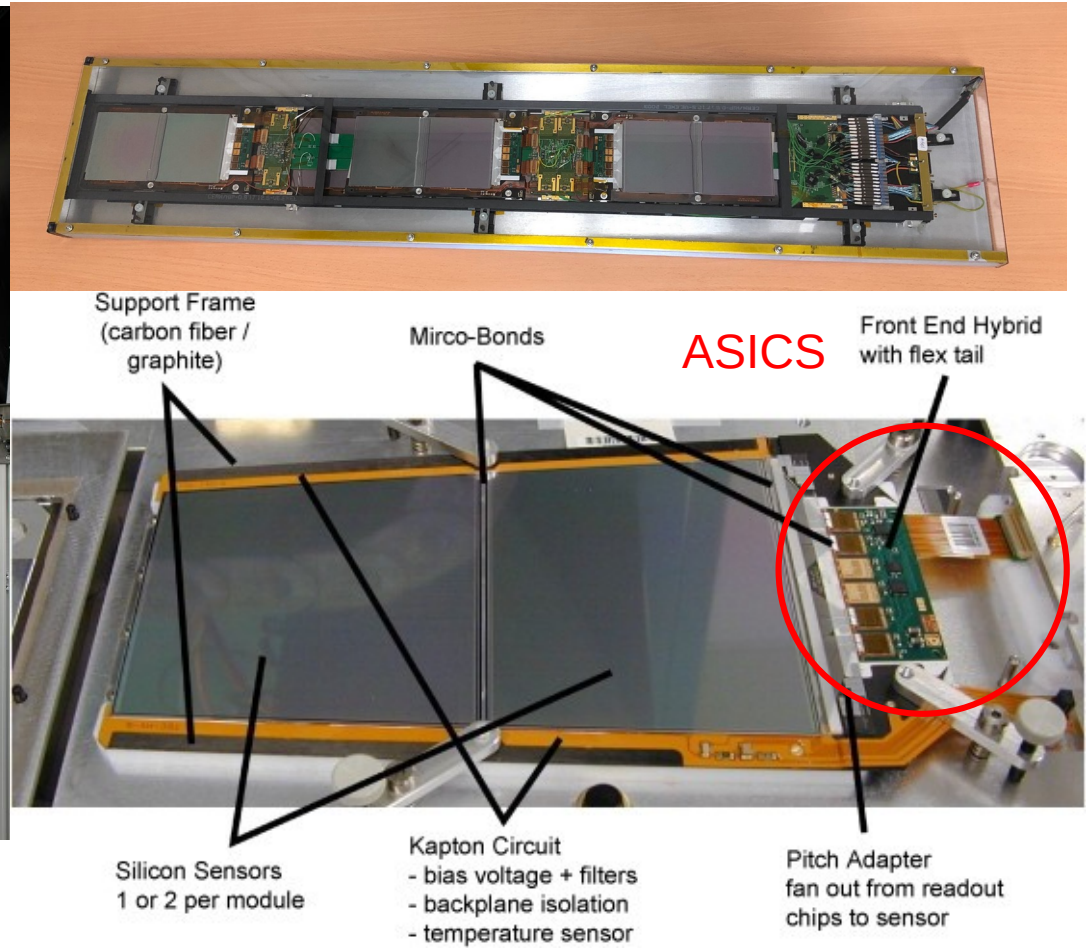


Considerations

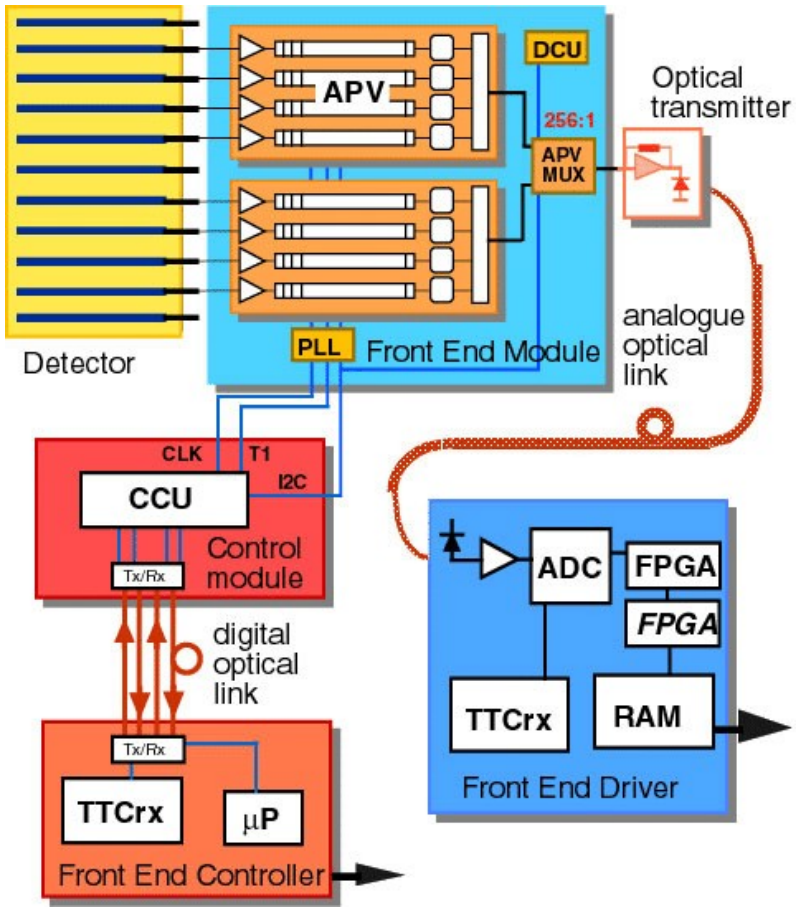
- Easy to add a new device
- Scalable, guaranteed bandwidth
- Congestion : latency usually not guaranteed

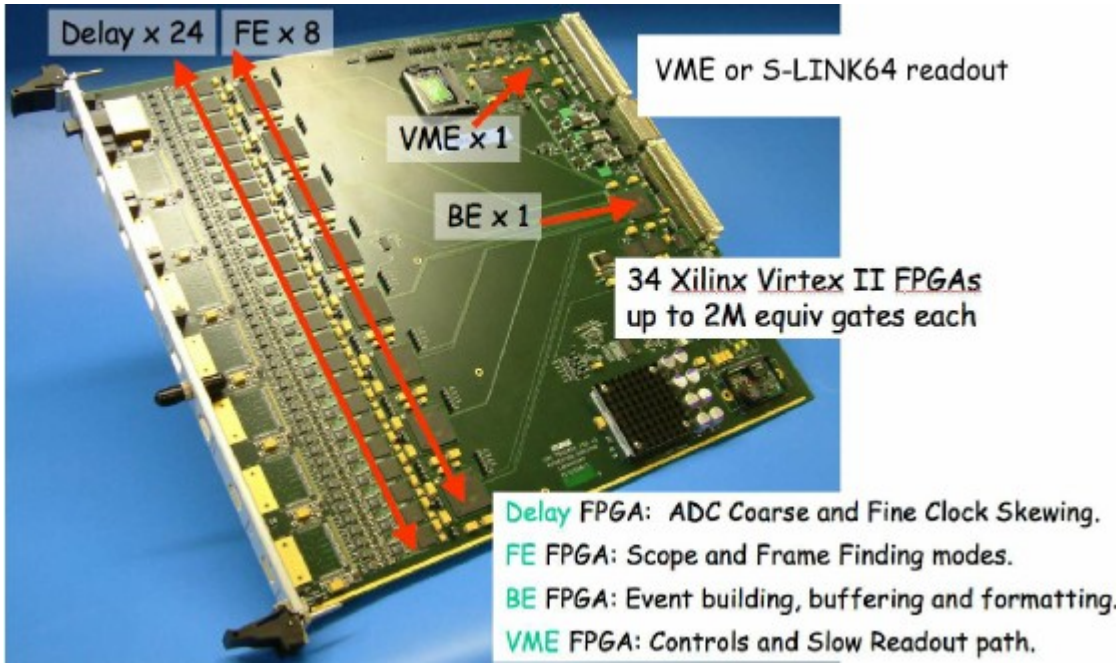
Implementation Example : CMS SST



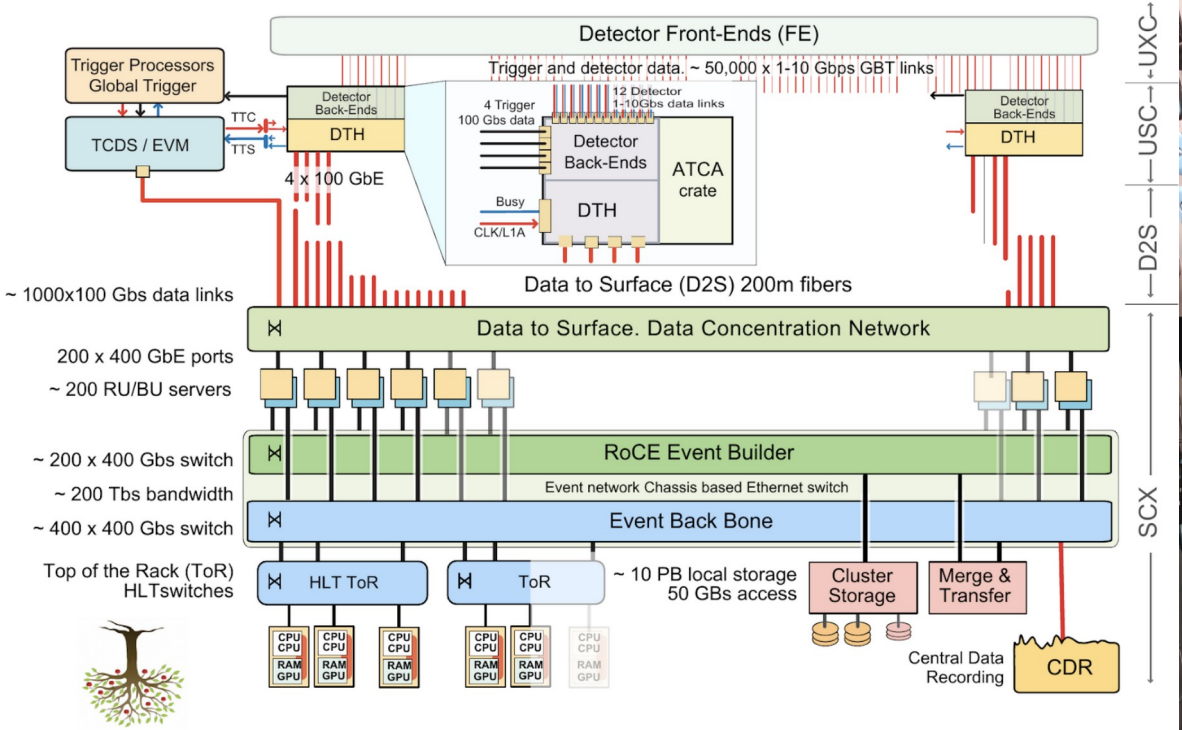


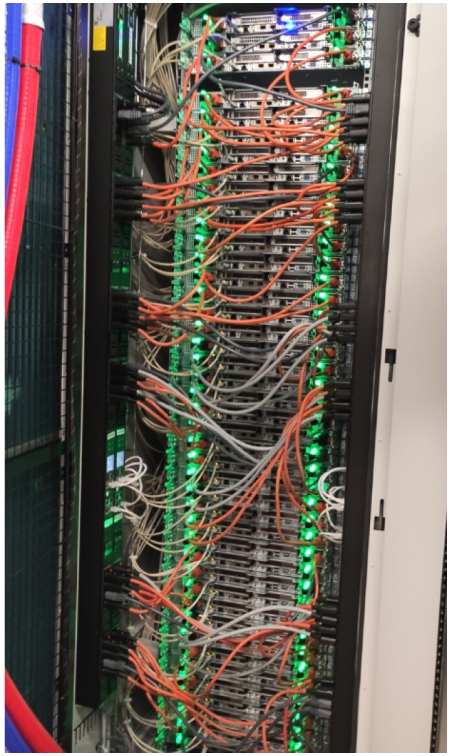
Implementation Example : CMS SST





Implementation Example : CMS



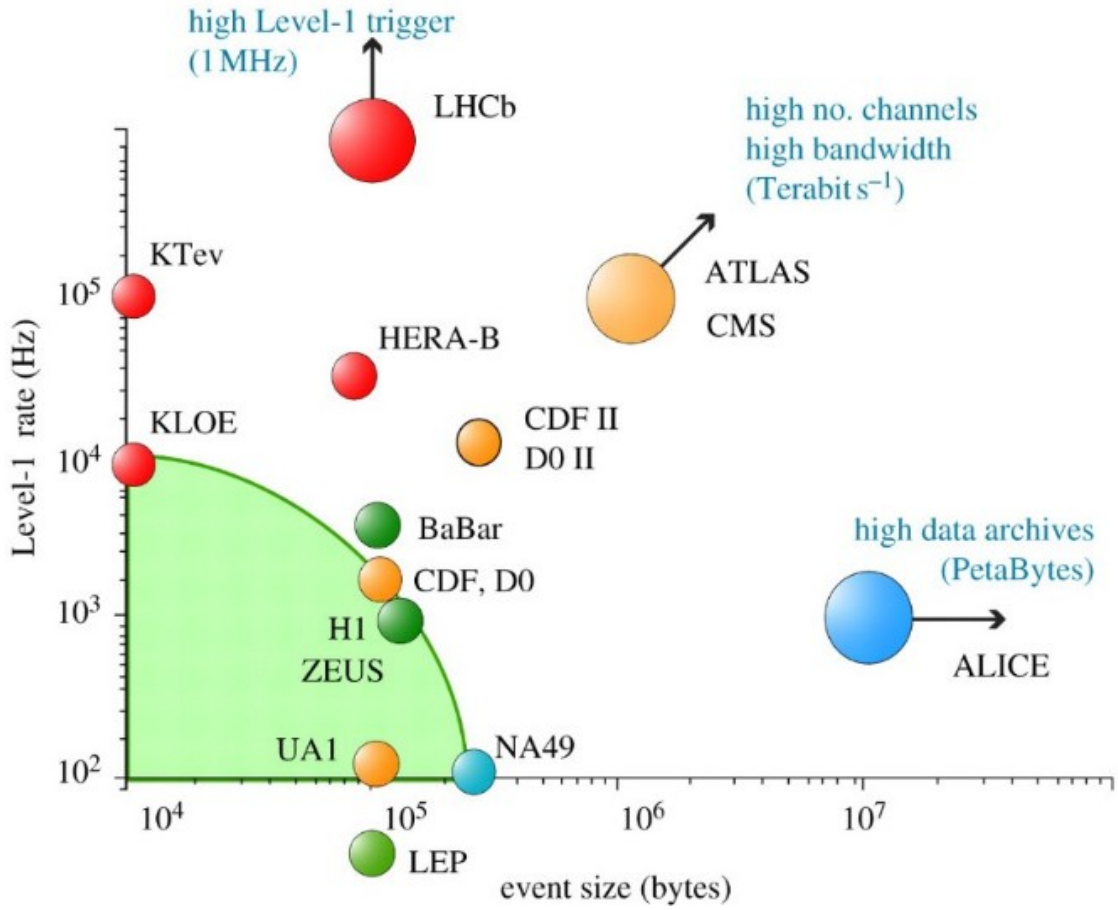


Large Scale DAQ : LHC

LHC experiments provide examples of large multi-tiered DAQ

- 10^6-10^8 channel counts
- O(MB) event sizes
- High data rates

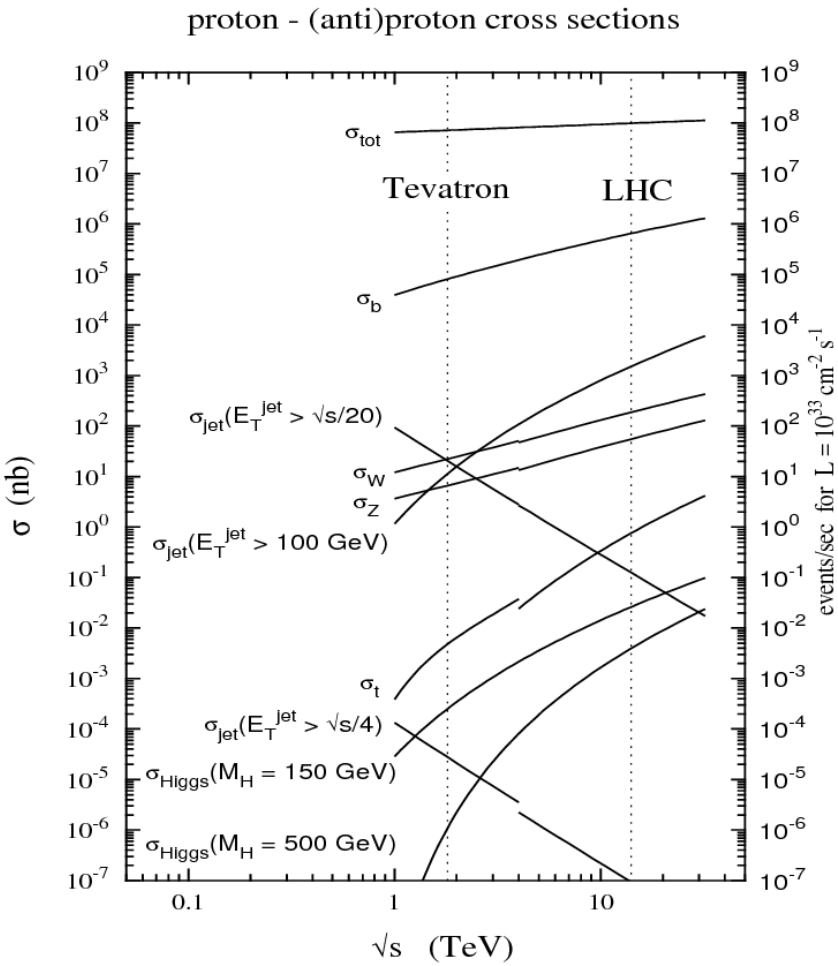
But interesting differences among them!



Large Scale DAQ : LHC

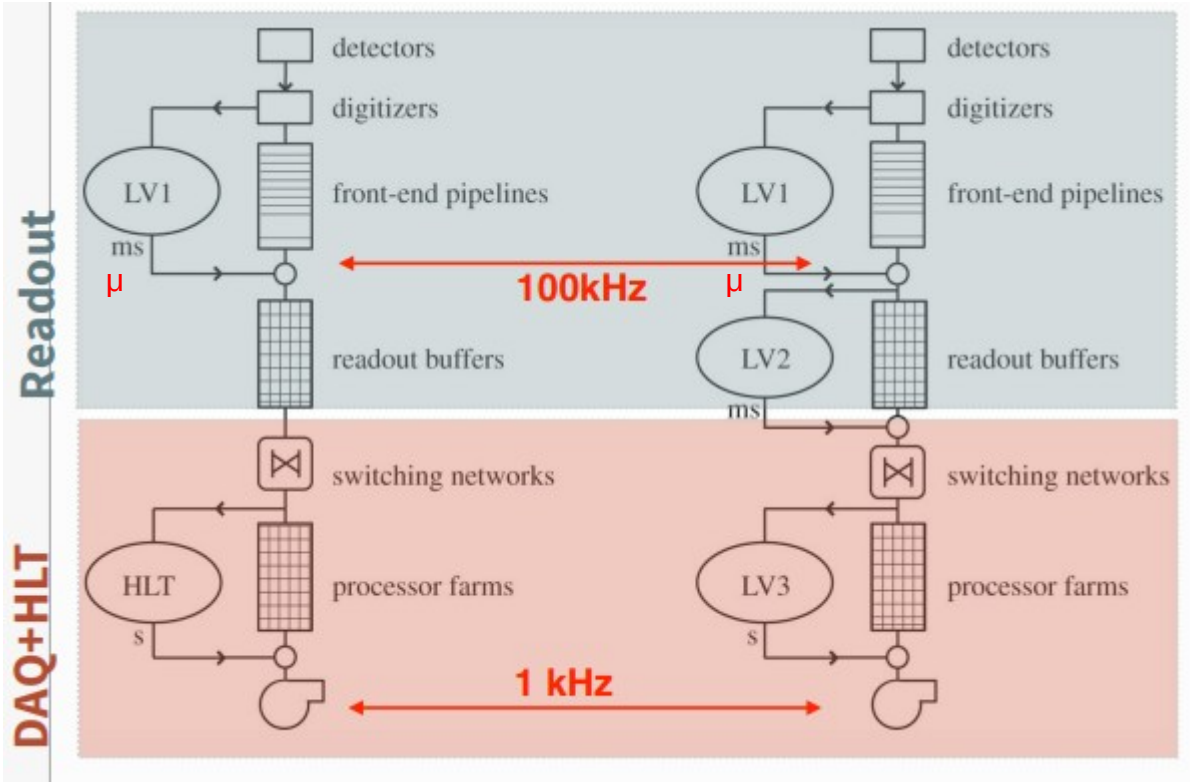
Physics based triggers

- Interesting physics has low cross section → need high luminosity!
- But uninteresting physics contributes with high cross section
- Can't store it all …
- Event filtering based on interesting physics signatures
  - Eg: High energy jets, leptons
- Multi-level event selections
  - Coarse with low latency in h/w
  - Particle-like with higher latency in s/w



Large Scale DAQ : LHC

Example : ATLAS & CMS trigger tiers



Hardware Level-1

- FPGA-based
- Detector readout upon accept
- Latency O(10) μs
- Accept rate ~100 kHz

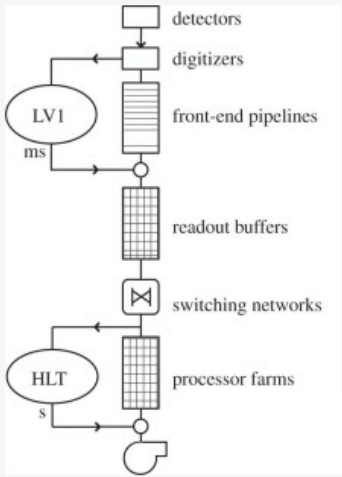
Software HLT

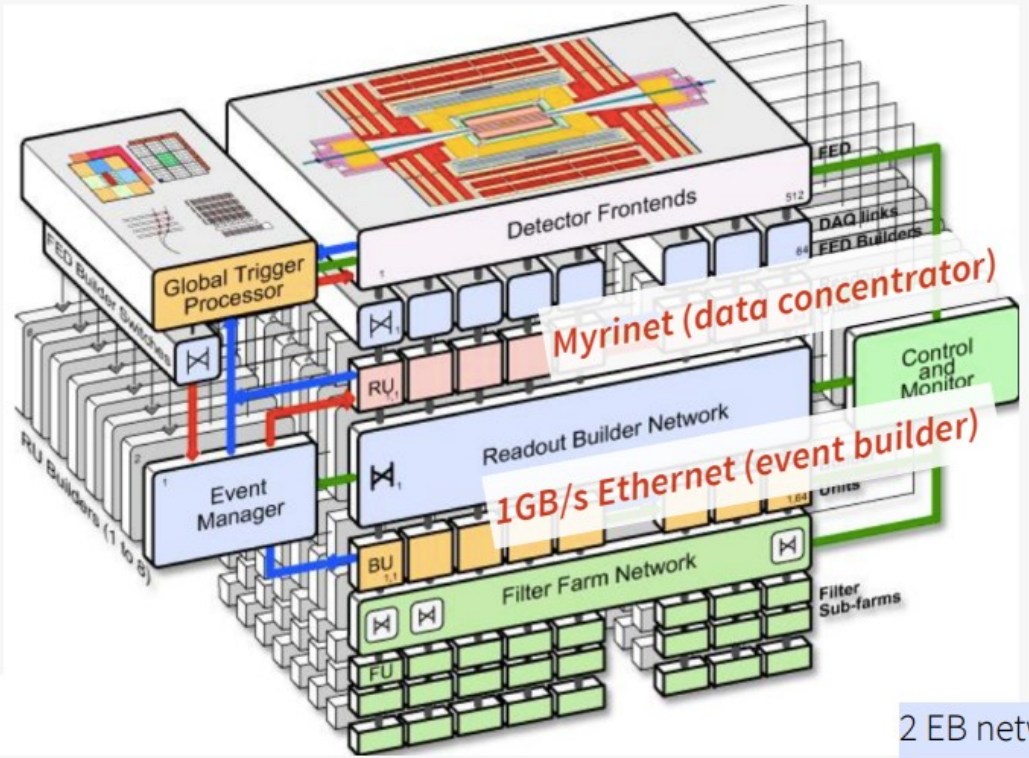
- Offline-like reconstruction
- 100 GB/s network,
- 10^4 cores
- Latency O(ms) ++
- 1 kHz accept rate

CMS Run-1 TDAQ

Run-1 (as from TDR, 2002)

- Myrinet + 1GBEthernet
- 1-stage building: 1200 cores (2C)
- HLT: ~13,000 cores
- 18 TB memory @100kHz:
  ~90ms/event

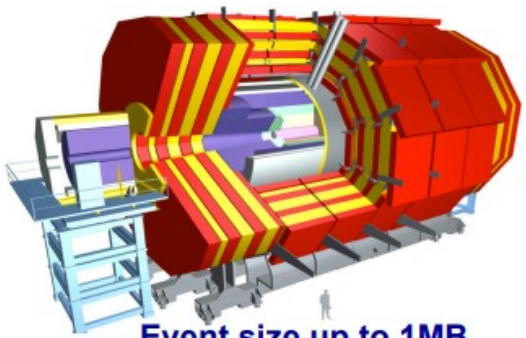




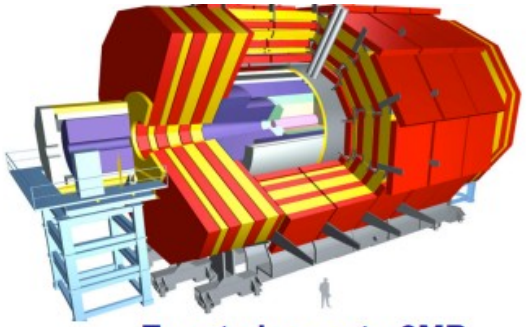
2 EB networks

Filter network

CMS Run-2/3 TDAQ

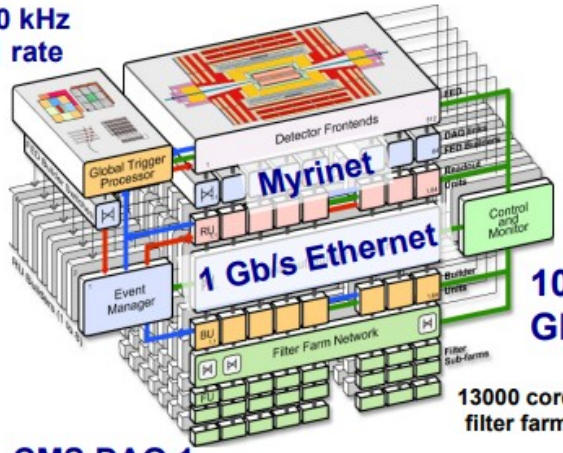


Event size up to 1MB



Event size up to 2MB
(large margin)

100 kHz
L1 rate

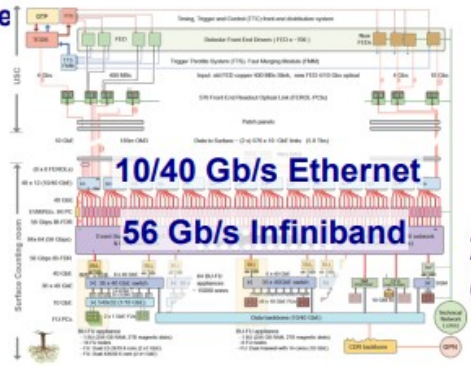


100
GB/s

13000 core
filter farm

CMS DAQ 1

100 kHz
L1 rate



200
GB/s

16000+ core
filter farm

CMS DAQ 2

max. 1.2 GB/s to storage

~ 3 GB/s to storage

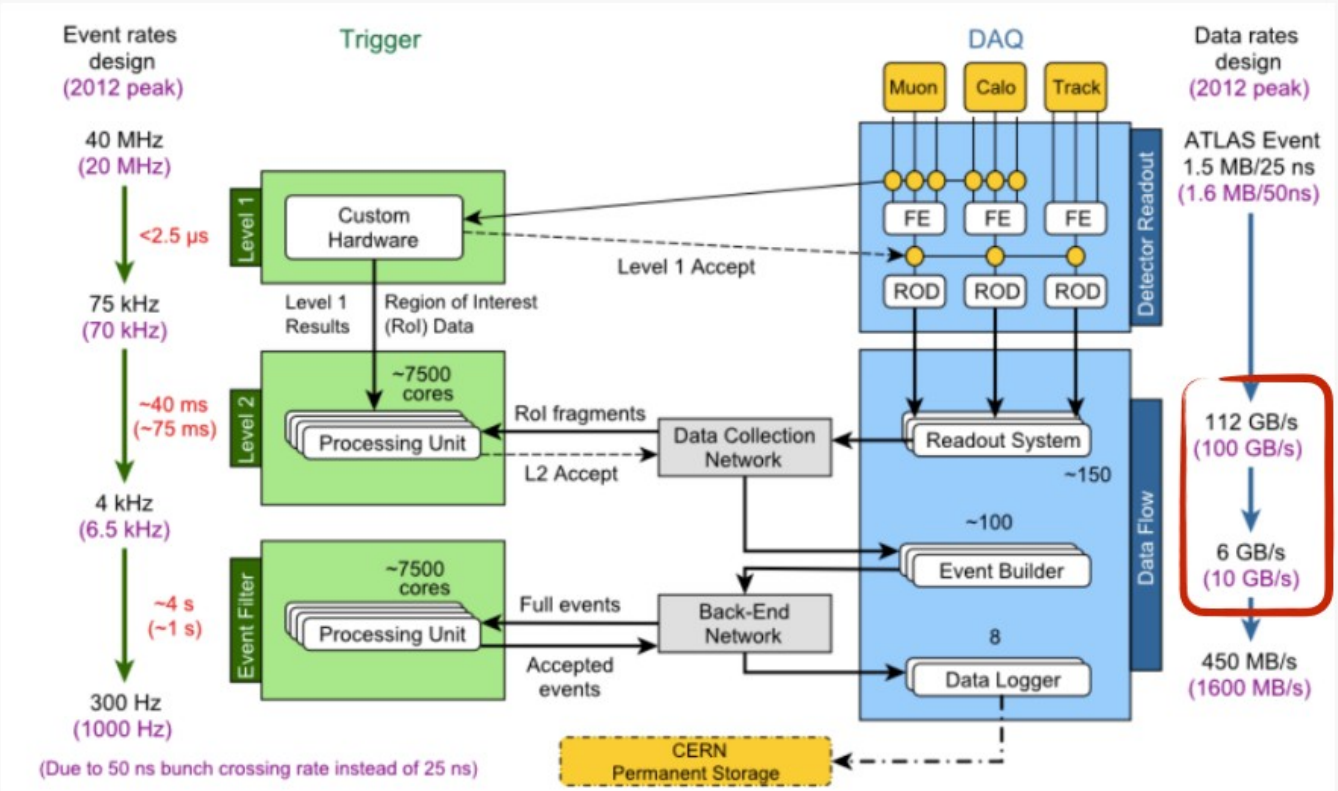
ATLAS Run-1 TDAQ

Overall network bandwidth: ~10 GB/s

- x10 reduced by regional readout)

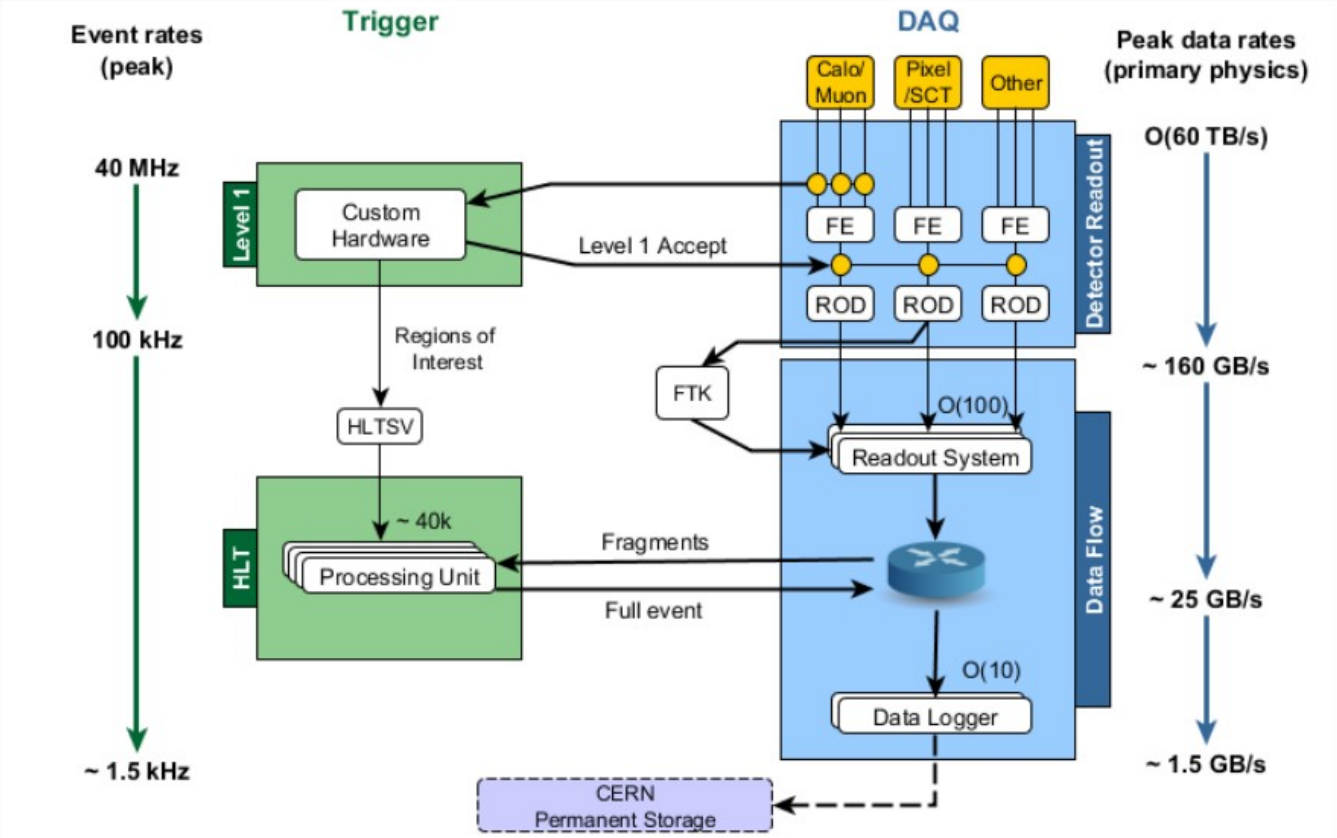
Complex data routing

HLT reconstruction
Seeded by Level-2
Regions of Interest

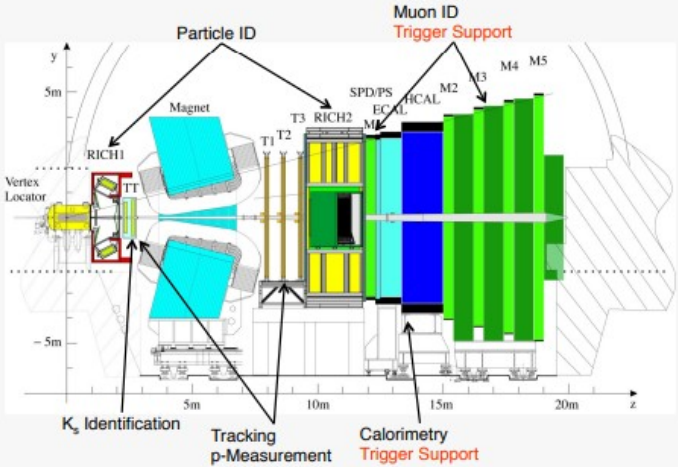


ATLAS Run-2/3 TDAQ

- Increased rates
- Merged L2/HLT
- Increase Readout bandwidth
- Increase HLT rate
- Unified network



LHCb Run-2 TDAQ

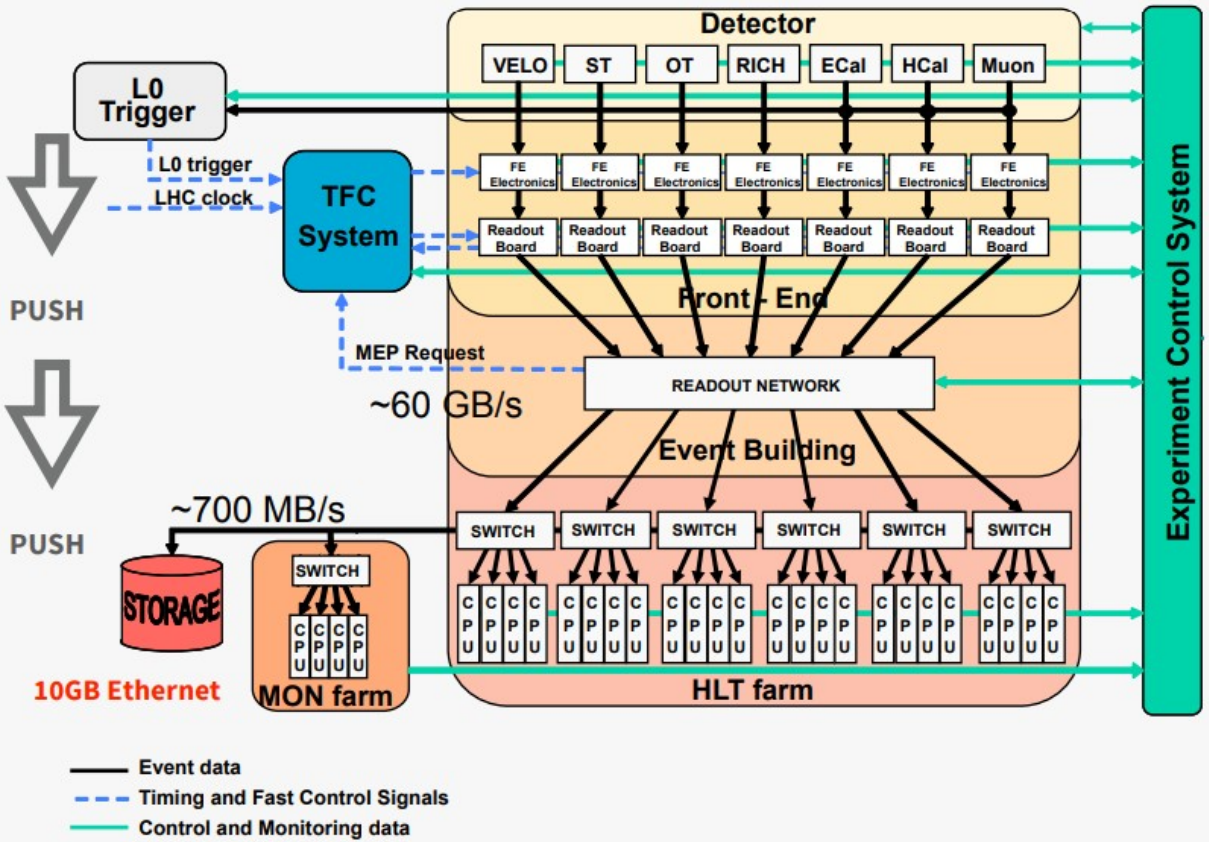


Single forward arm spectrometer → reduced event size

- Average event size 60 kB
- Average rate into farm 1 MHz
- Average rate to tape ~12 kHz

Small event, at high rate

- optimised transmission



LHCb Run-3 TDAQ

(Level-1) Trigger-less!

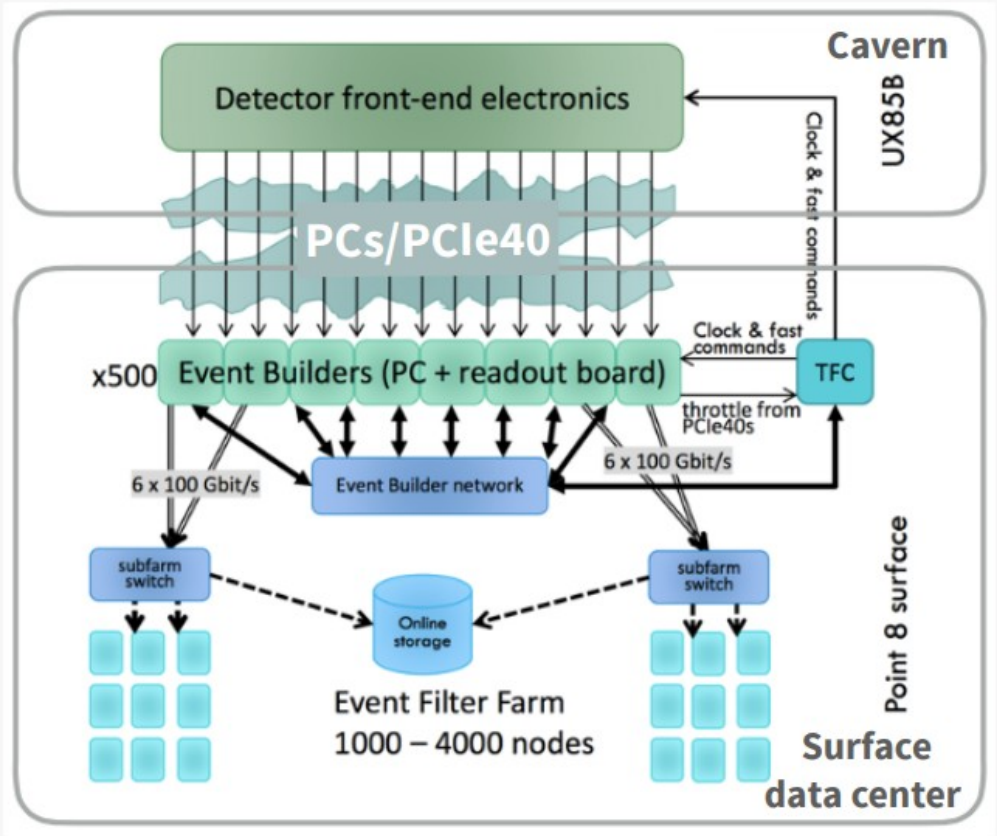
Data reduction before EB

- custom readout FPGA-card (PCIe40)
- Each sub detectors with its packing algorithm, i.e. zero-suppression and clustering

Readout: ~10,000 GBT links (4.8 Gb/s, rad-hard)

DataFlow: decouple EB and HLT in 2 networks

- scalable up to 400 x 100Gbps links



GPU selection (HLT1) colocated with PCIe event building

Summary

A short introduction to DAQ

- From basic ingredients : trigger, buffers, back-pressure
- Then scaling from few to many channels
  - Multi-tier DAQ, data transfer
- Ending with full implementations
  - Using LHC experiments as the example

Summary

A short introduction to DAQ

- From basic ingredients : trigger, buffers, back-pressure
- Then scaling from few to many channels
  - Multi-tier DAQ, data transfer
- Ending with full implementations
  - Using LHC experiments as the example

The magic animating our experiments!

Summary

A short introduction to DAQ

- From basic ingredients : trigger, buffers, back-pressure
- Then scaling from few to many channels
  - Multi-tier DAQ, data transfer
- Ending with full implementations
  - Using LHC experiments as the example

Much more to cover than time permits!

- Techniques and protocols, control paths, timing & synch
- Local DAQ : detector-specific procedures for acquiring sensible data
  - Noise scans, signal injection tests, calibrations …
- TDAQ for future experiments