THE UNIVERSITY of EDINBURGH

April 15 – 19, 2024
Edinburgh, United Kingdom

12th Beam Telescopes and Test Beams Workshop

INTRODUCTION TO TDAQ AND ITS SCALING PRINCIPLES

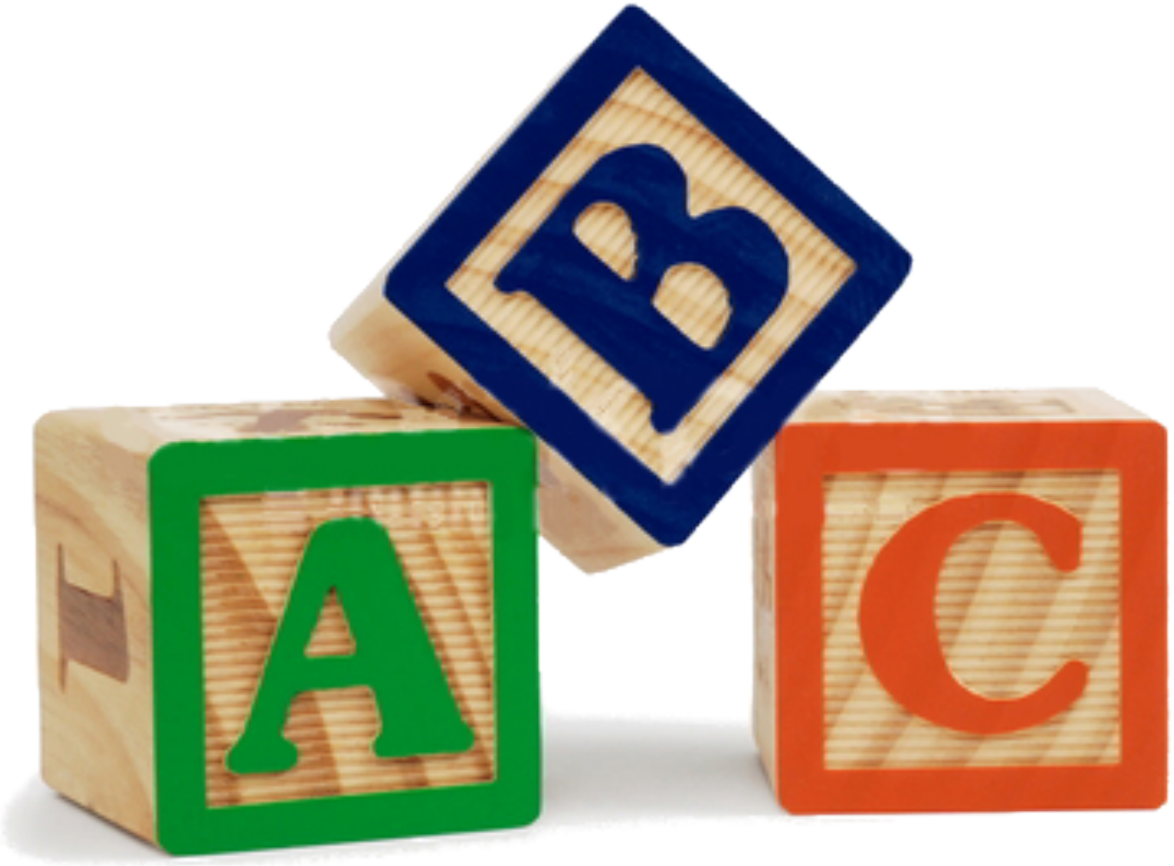
F.Pastore (Royal Holloway Univ. of London)
francesca.pastore@cern.ch

ROYAL HOLLOWAY UNIVERSITY OF LONDON

INTRODUCTION

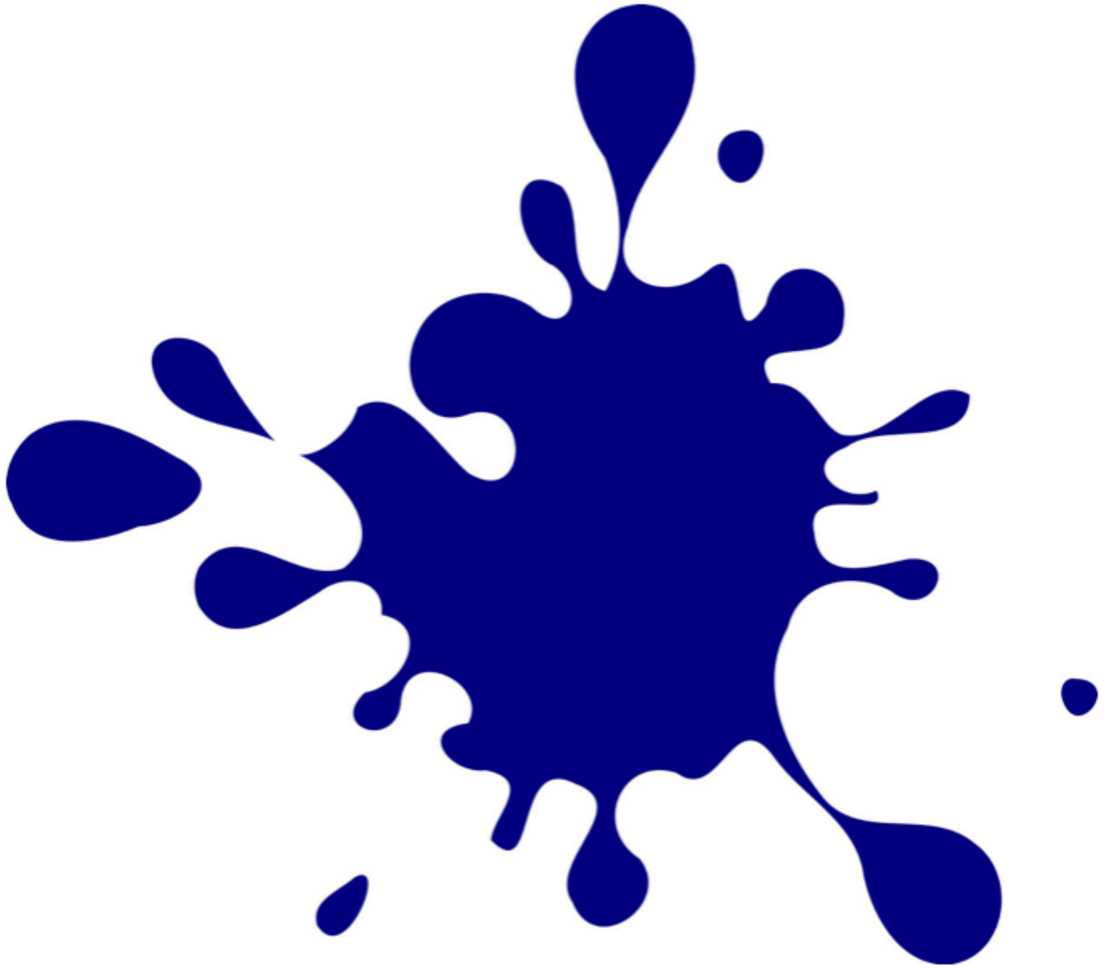
- **Aim of this lecture is to introduce the basic TDAQ concepts, avoiding as many technological details as possible**
- **Focus on High Energy Physics**
  - But key concepts are common to other areas

Credits to A.Negri and W.Vandelli whose material helped me preparing these slides



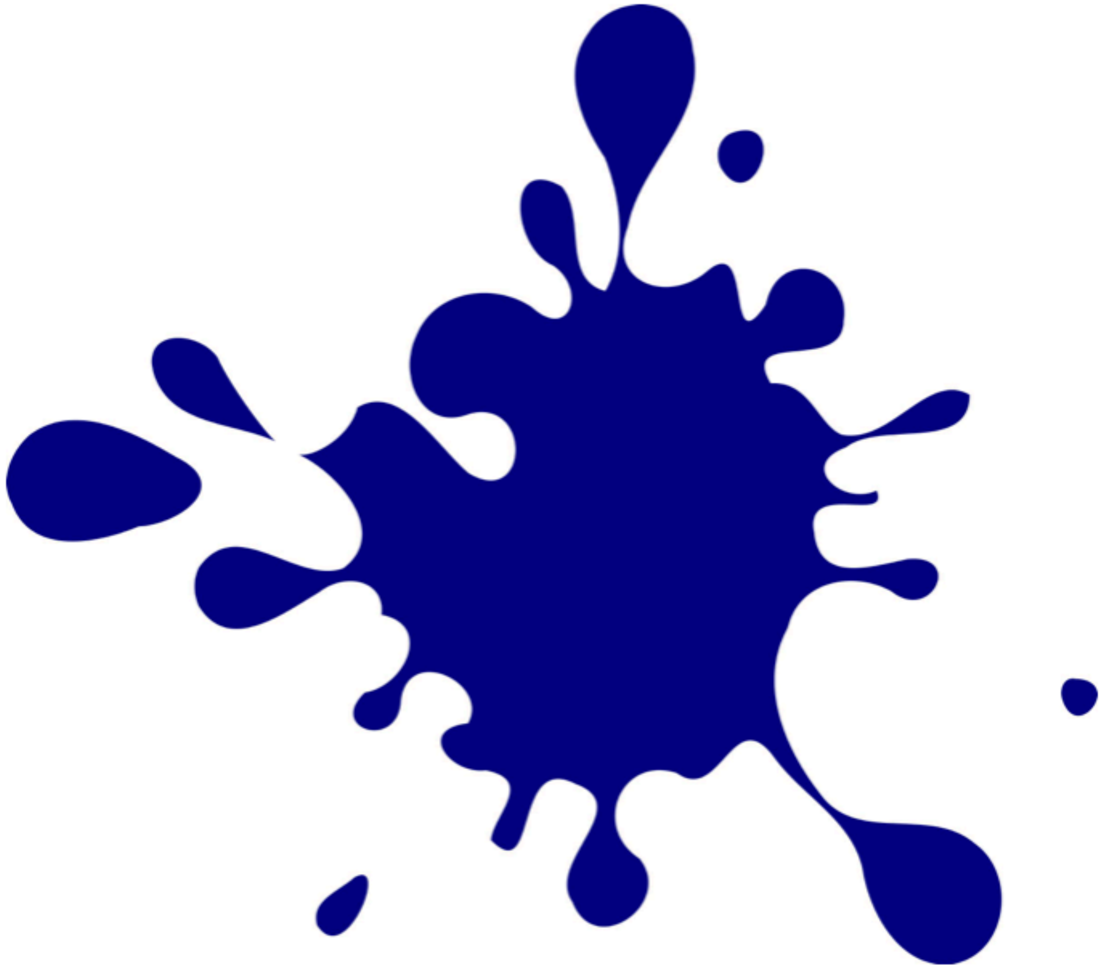
OUTLINE

- **Introduction**
  - What is Trigger and DAQ?
  - Overall TDAQ framework
- **Basic TDAQ concepts**
  - Digitization, Latency
  - Deadtime, Busy
  - De-randomization
- **Scaling up**
  - Readout and Event Building
  - Buses vs Network
- **Fight bottlenecks**



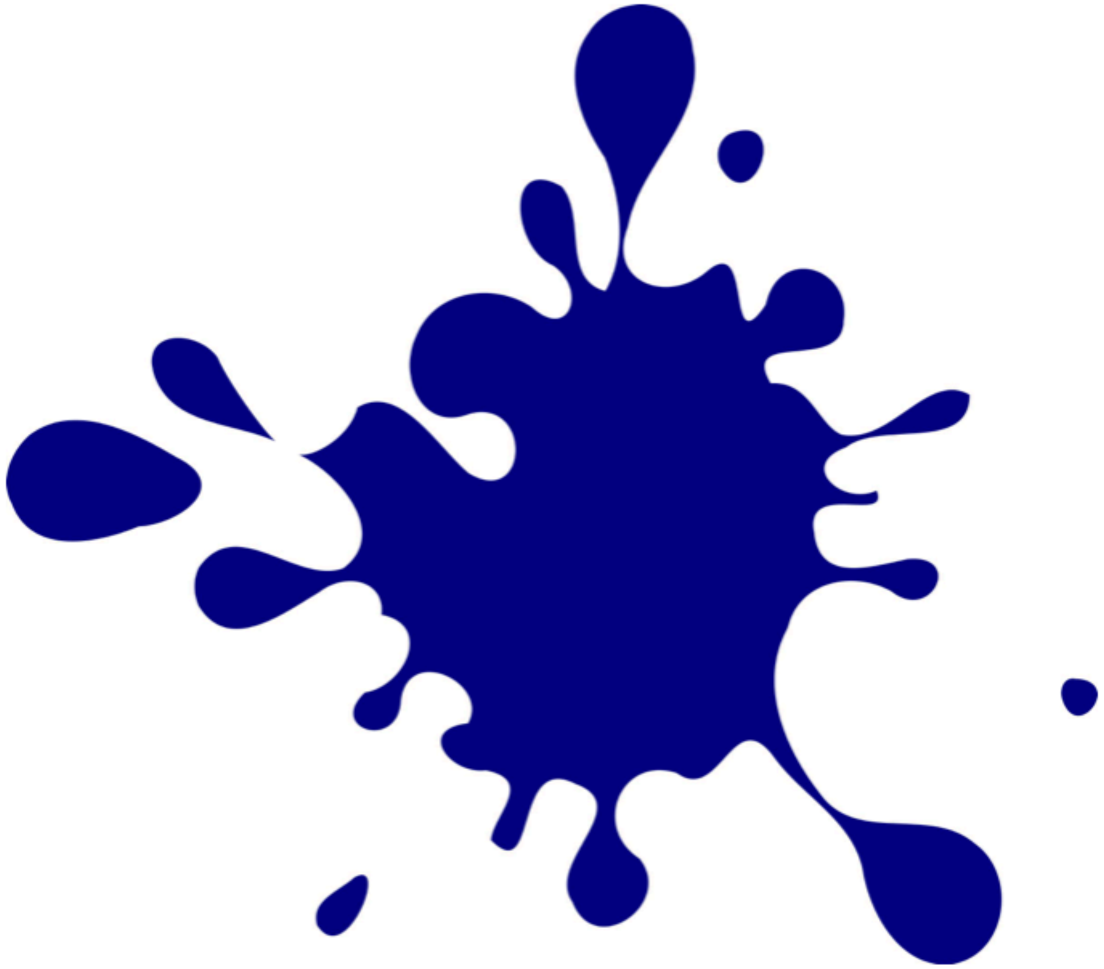
OUTLINE

- **Introduction**
  - What is Trigger and DAQ?
  - Overall TDAQ framework
- **Basic TDAQ concepts**
  - Digitization, Latency
  - Deadtime, Busy
  - De-randomization
- **Scaling up**
  - Readout and Event Building
  - Buses vs Network
- **Fight bottlenecks**

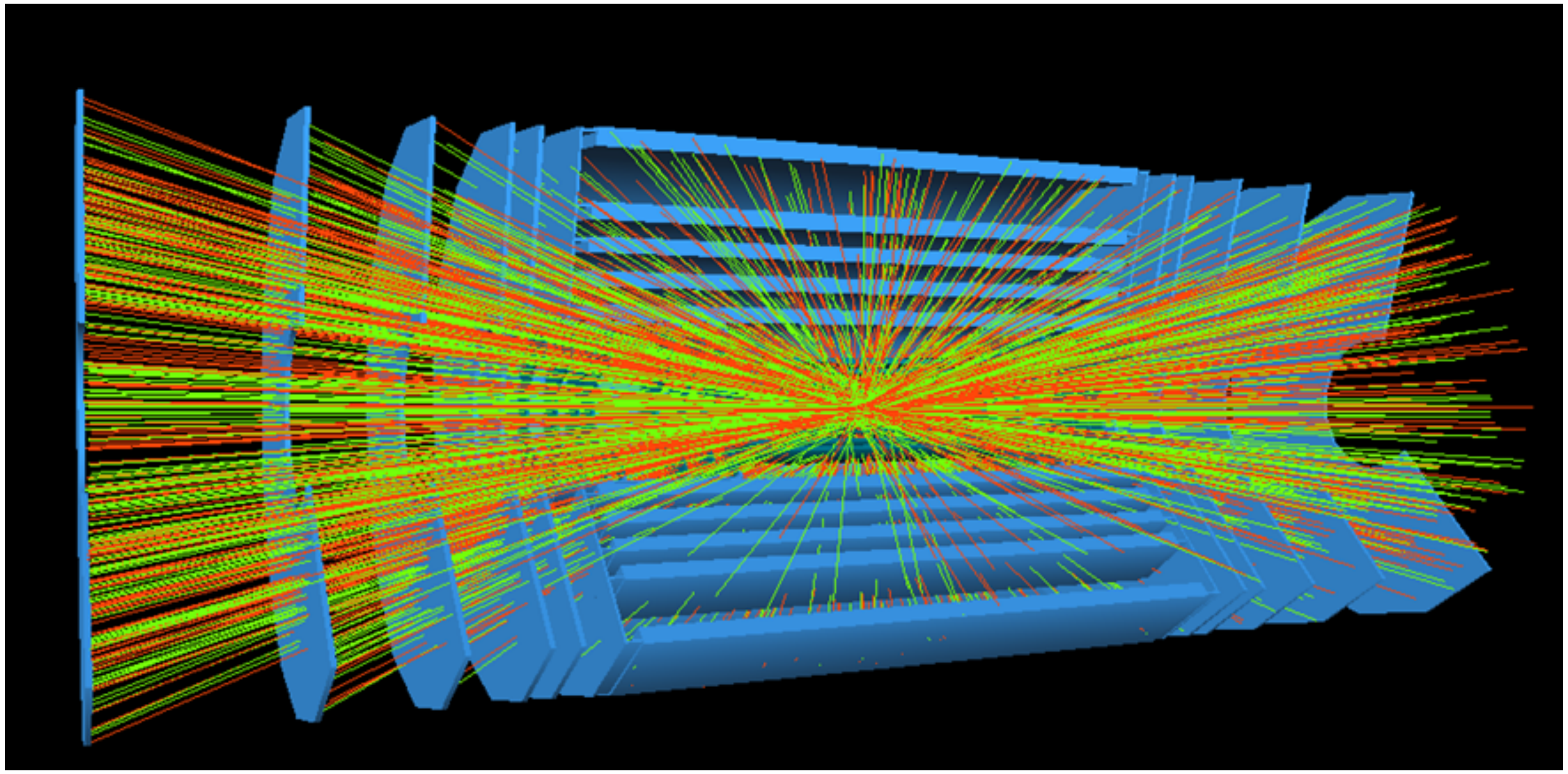


OUTLINE

- **Introduction**
  - What is Trigger and DAQ?
  - Overall TDAQ framework
- **Basic TDAQ concepts**
  - Digitization, Latency
  - Deadtime, Busy
  - De-randomization
- **Scaling up**
  - Readout and Event Building
  - Buses vs Network
- **Fight bottlenecks**



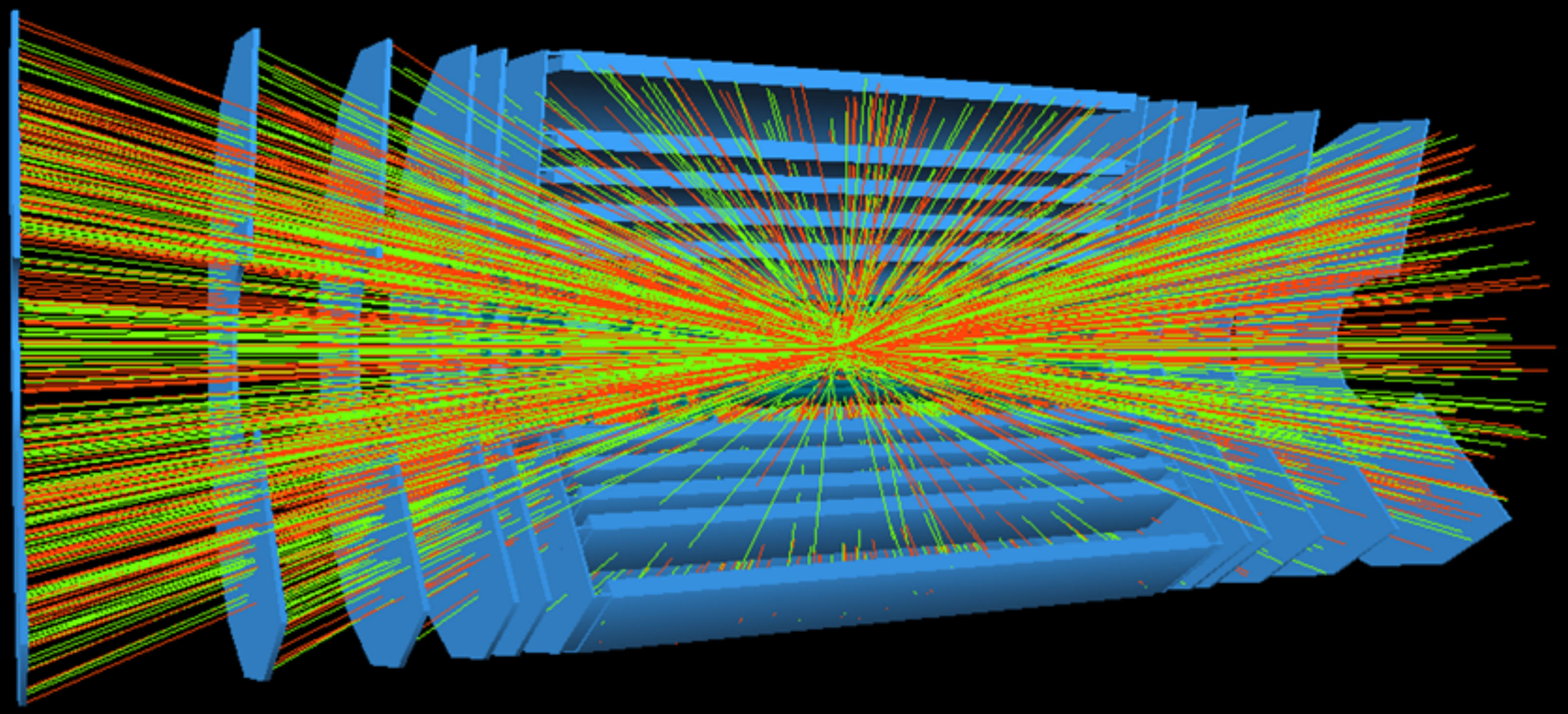
WHAT IS DAQ?



[Wikipedia]

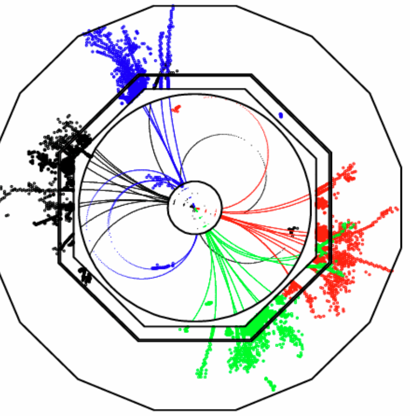
- ➡ **Data AcQuisition (DAQ) is**
  - ➡ the process of sampling signals that measure real world physical conditions
  - ➡ and converting the resulting samples into digital numeric values that can be manipulated by a PC
- ➡ **Main role of DAQ in HEP**
  - ➡ process the signals generated in a detector
  - ➡ and saving the (interesting) information on a permanent storage

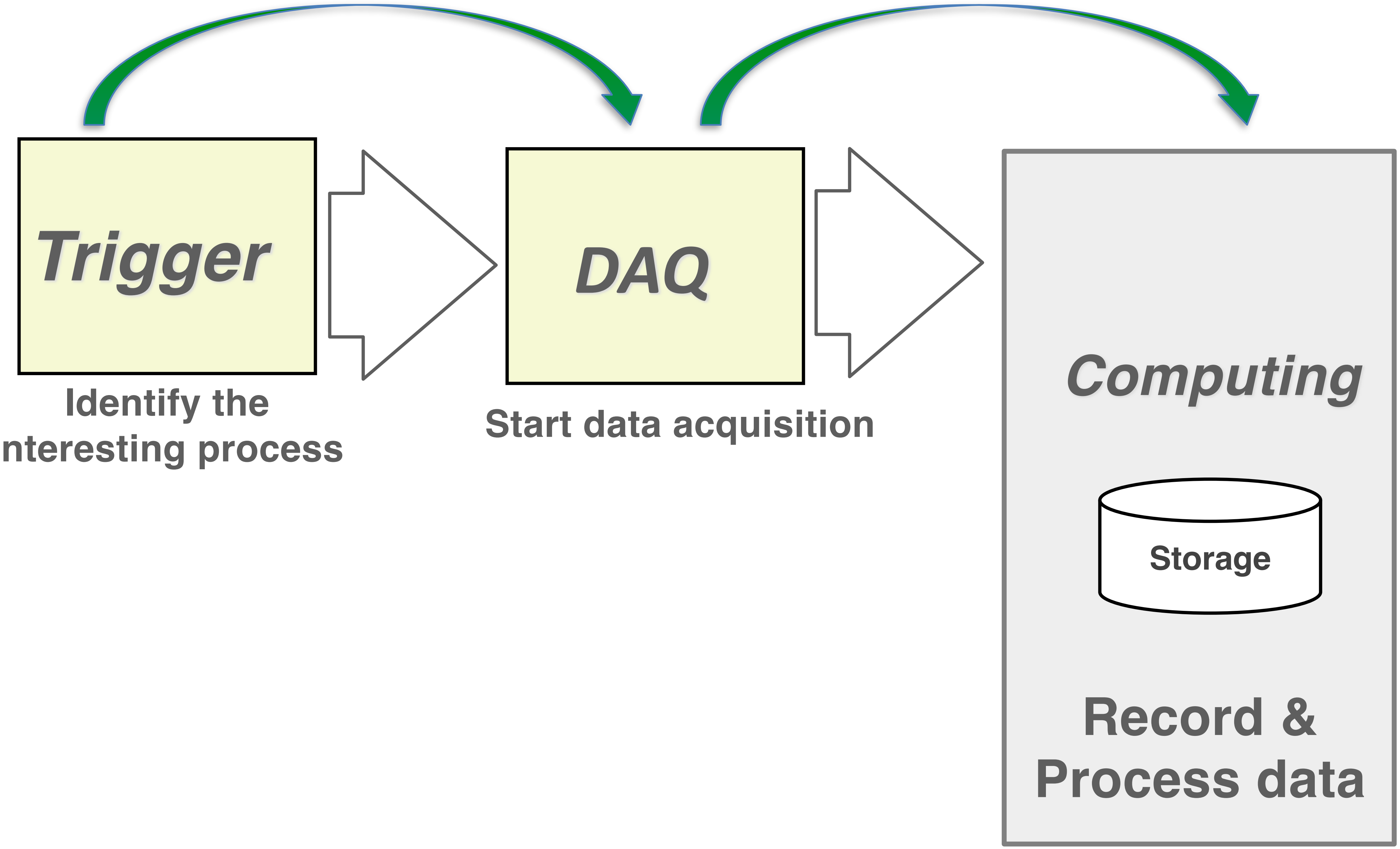
THE DATA DELUGE



- **In many systems, like particle physics or astronomy experiments, to store all the possibly relevant data provided by the sensors is UNREALISTIC and often becomes also UNDESIRABLE**
- **Three approaches are possible:**
  - Reduced amount of data (packing and/or filtering) →***Trigger!***
  - Faster data transmission and processing
  - Both!

LINK BETWEEN TRIGGER AND DAQ





limited by time constraints

limited by computing resources

Trigger

Identify the interesting process

DAQ

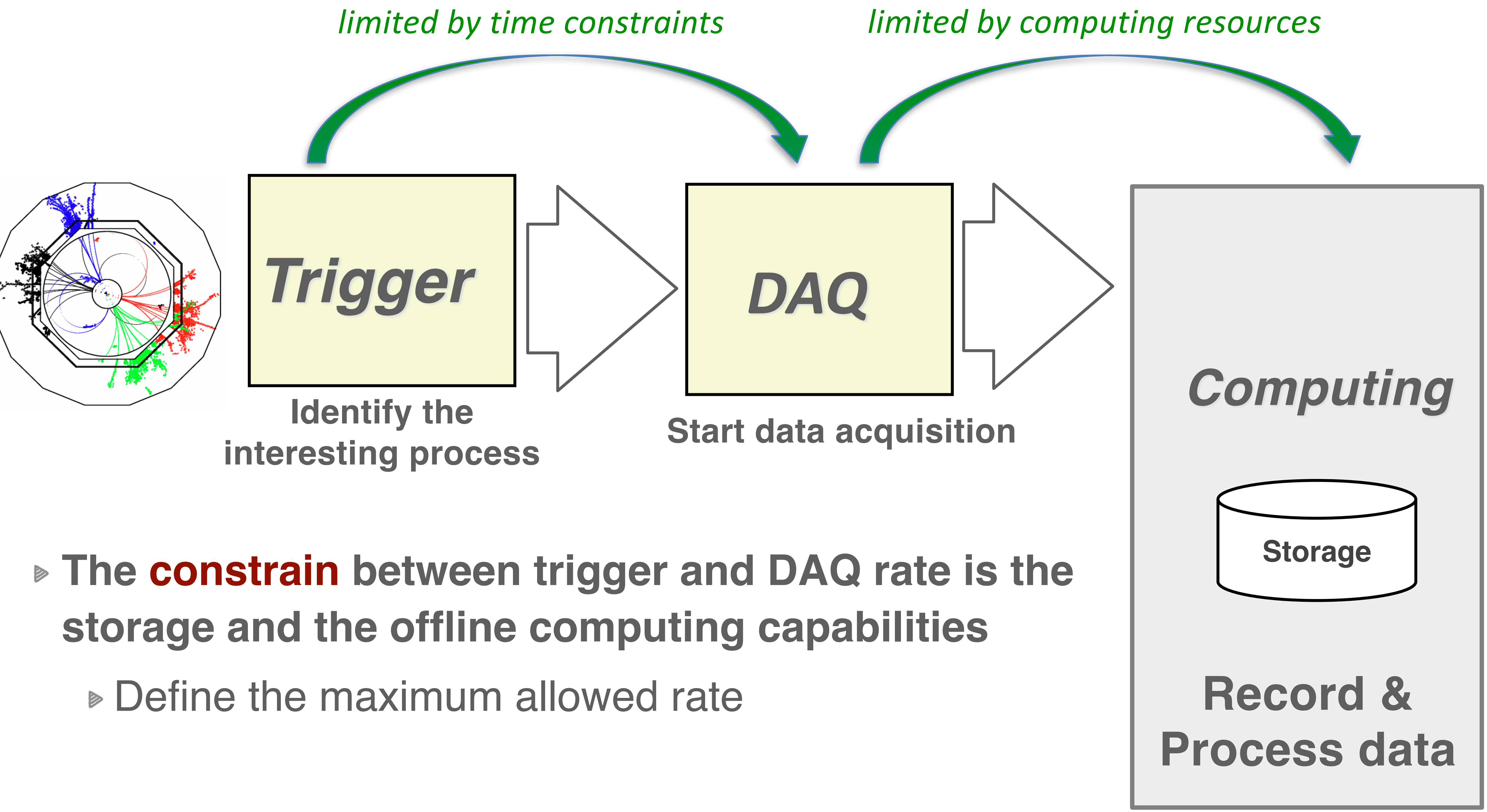
Start data acquisition

Computing

Storage

Record & Process data

LINK BETWEEN TRIGGER AND DAQ



limited by time constraints

limited by computing resources

Trigger

Identify the interesting process

DAQ

Start data acquisition

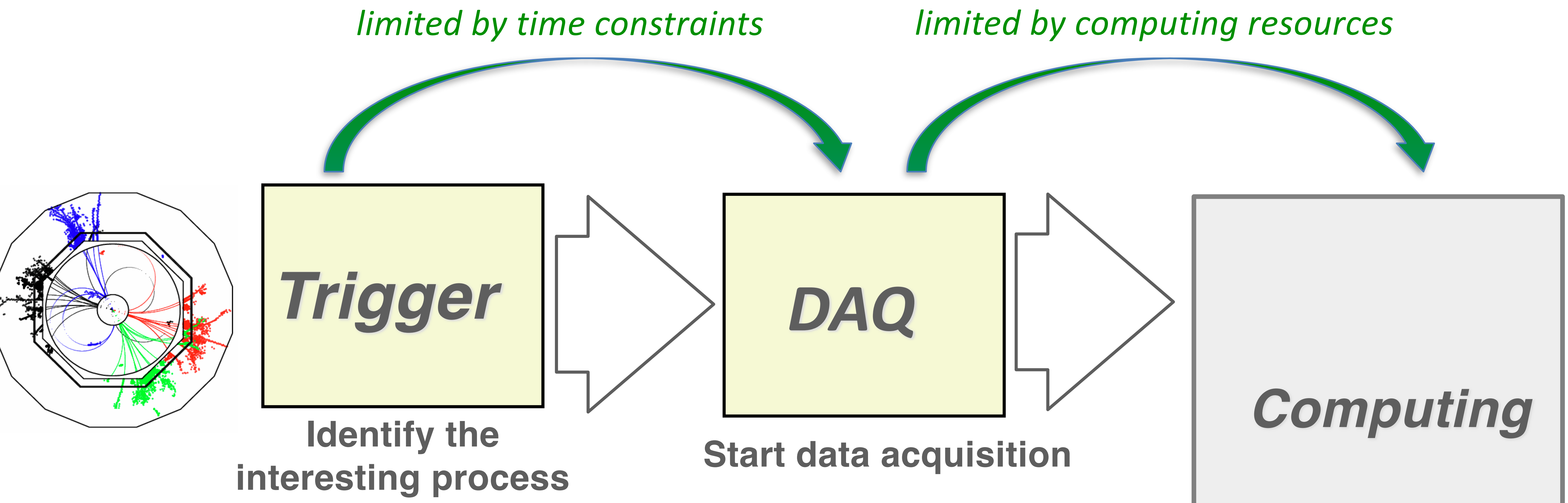
Computing

Storage

Record & Process data

- **The** **constrain** **between trigger and DAQ rate is the storage and the offline computing capabilities**
  - Define the maximum allowed rate

LINK BETWEEN TRIGGER AND DAQ



limited by time constraints

limited by computing resources

Trigger

Identify the interesting process

DAQ

Start data acquisition

Computing

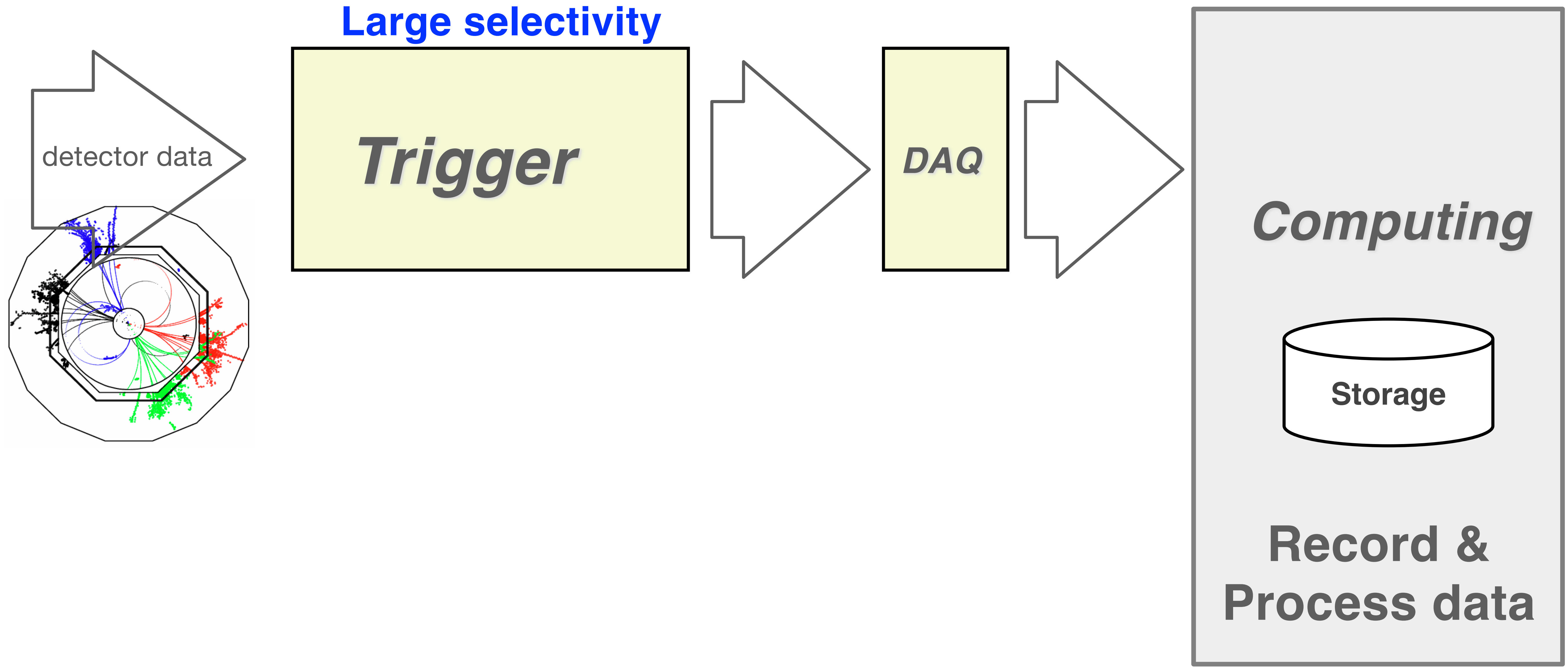
Storage

Record & Process data

- **The constrain between trigger and DAQ rate is the storage and the offline computing capabilities**
  - Define the maximum allowed rate
- **Which is the balance between Trigger and DAQ resources?**

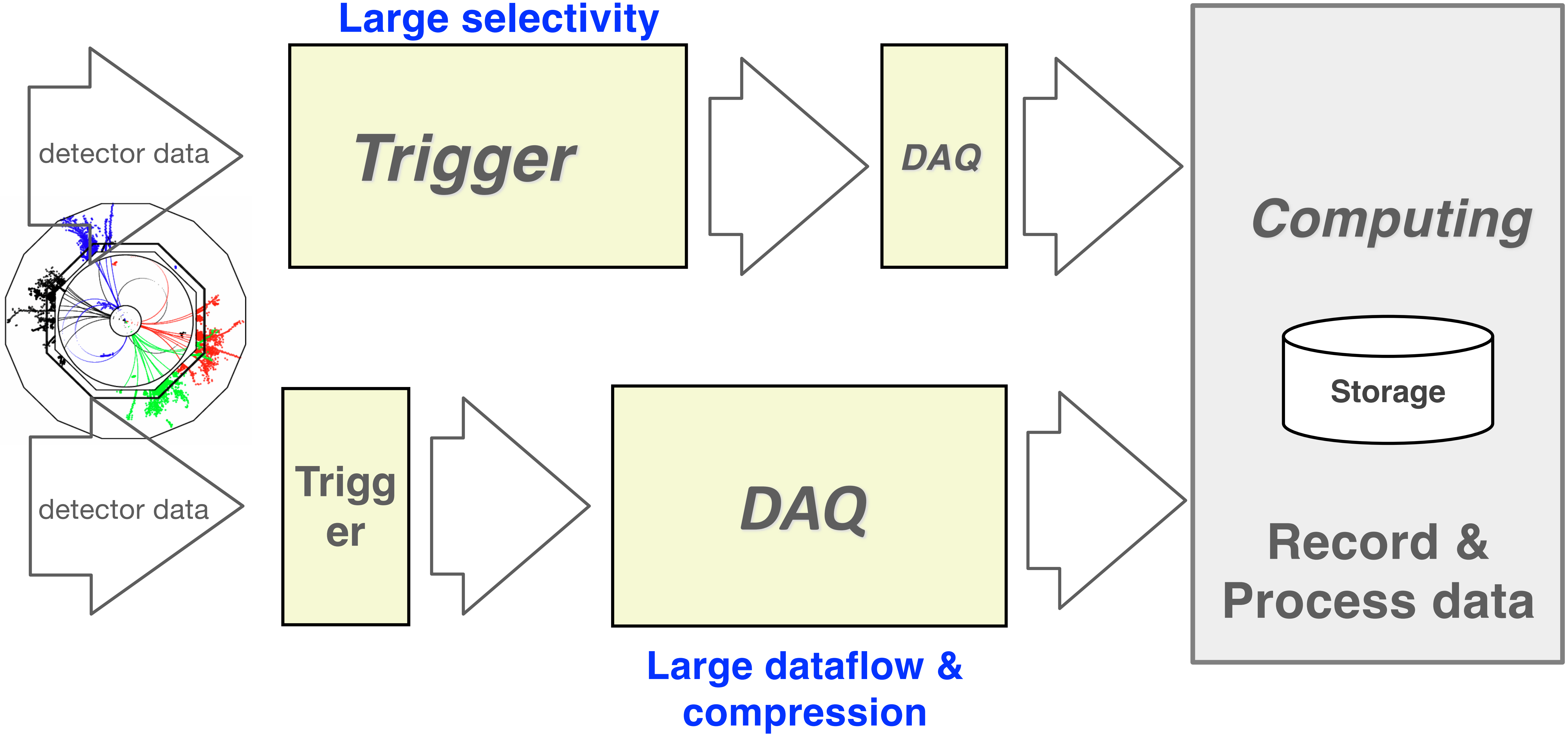
BALANCE BETWEEN TRIGGER AND DAQ

- If the trigger is highly selective, one can reduce the size of the dataflow



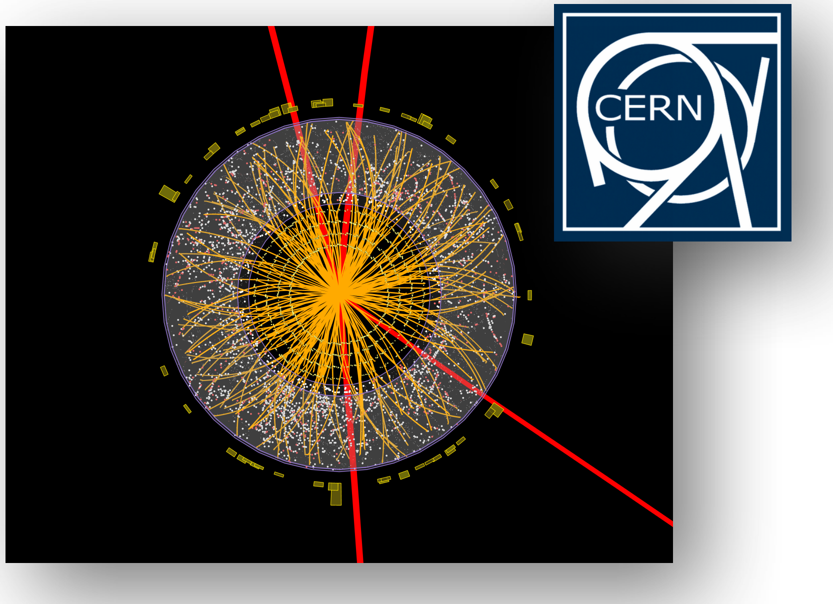
BALANCE BETWEEN TRIGGER AND DAQ

- If the trigger is highly selective, one can reduce the size of the dataflow



- If the selectivity of the trigger is not enough, due to large irreducible background, a large data flow (and data compression) is needed

TWO OPPOSITE EXAMPLES



- **LHC – ATLAS**

- Project started in 1996
- Technology chosen in 2000
- Start data-taking 2008

- Full p-p collision rate: 40 MHz
- Average event size: 1.5 MB
- Full data rate: ~60 PB/s

- Defined physics signal
- Selective trigger reduces 7 orders of magnitudes to ~kHz

- Affordable DAQ rate: ~GB/s
- Data distribution (GRID)

- **SKA (Square Km Array)**

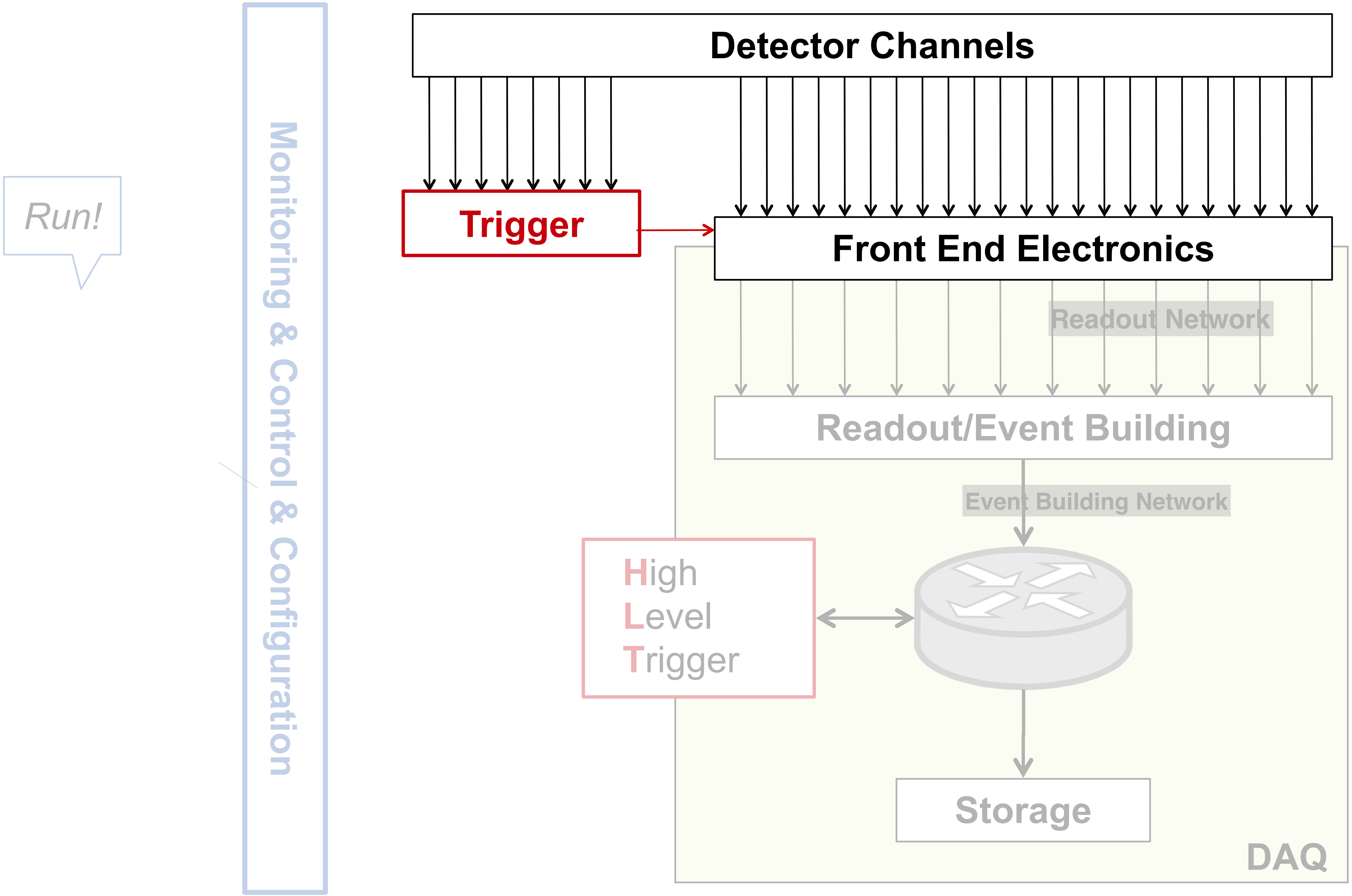
- Project started in 2011
- Technologies under evaluation
- Start operations in 2028

- Radio-photograph the sky continuously
- 1.12 PB/s of photos collected

- EXASCALE system: 10^{18} operations for correlation and imaging
- Simple correlator : 10 TB/s
- Total Internet Traffic ≈ 8 TB/s in 2010

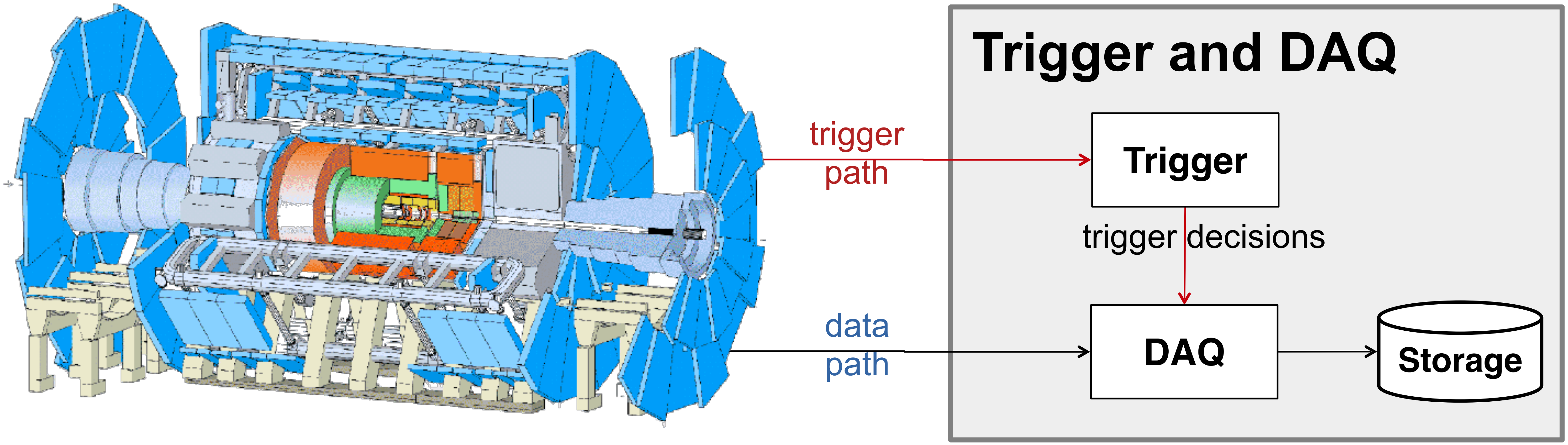
- Required large computing power
- Big-data and cloud-computing drive market

T/DAQ ARCHITECTURE



DOUBLE PATHS

- **Trigger path**
  - From dedicated detectors to trigger logic
  - online selection
- **Data path**
  - From all the detectors to storage
  - On positive trigger decision



Trigger and DAQ

trigger path

Trigger

trigger decisions

data path

DAQ

Storage

TRIGGER: A REAL-TIME FILTER

- ➡ **Use discriminating features within widely extended systems**
- ➡ **Reality is:**
  - ➡ The trigger accepts events with features similar to the signal
  - ➡ The final rate is often dominated by not interesting physics

TRIGGER DUTIES

TRIGGER DUTIES

- ➡ **Either selects interesting events or rejects boring ones, in real time**
  - ➡ Selective: **efficient** for “signal” and **resistant** to “background”
  - ➡ Simple and robust
  - ➡ Quick

TRIGGER DUTIES

- **Either selects interesting events or rejects boring ones, in real time**
  - Selective: **efficient** for “signal” and **resistant** to “background”
  - Simple and robust
  - Quick
- **With minimal controlled latency**
  - time it takes to form and distribute its decision

TRIGGER DUTIES

- **Either selects interesting events or rejects boring ones, in real time**
  - Selective: **efficient** for “signal” and **resistant** to “background”
  - Simple and robust
  - Quick
- **With minimal controlled latency**
  - time it takes to form and distribute its decision
- **Generates a prompt signal used to start the data-acquisition processes**
  - To be distributed to front end electronics

TRIGGER DUTIES

- ➡ **Either selects interesting events or rejects boring ones, in real time**
  - ➡ Selective: **efficient** for “signal” and **resistant** to “background”
  - ➡ Simple and robust
  - ➡ Quick

- ➡ **With minimal controlled latency**
  - ➡ time it takes to form and distribute its decision
- ➡ **Generates a prompt signal used to start the data-acquisition processes**
  - ➡ To be distributed to front end electronics
- ➡ **Trigger and Front-End electronics have common design**
  - ➡ Data compression and formatting
  - ➡ Monitor and automatic fault detection

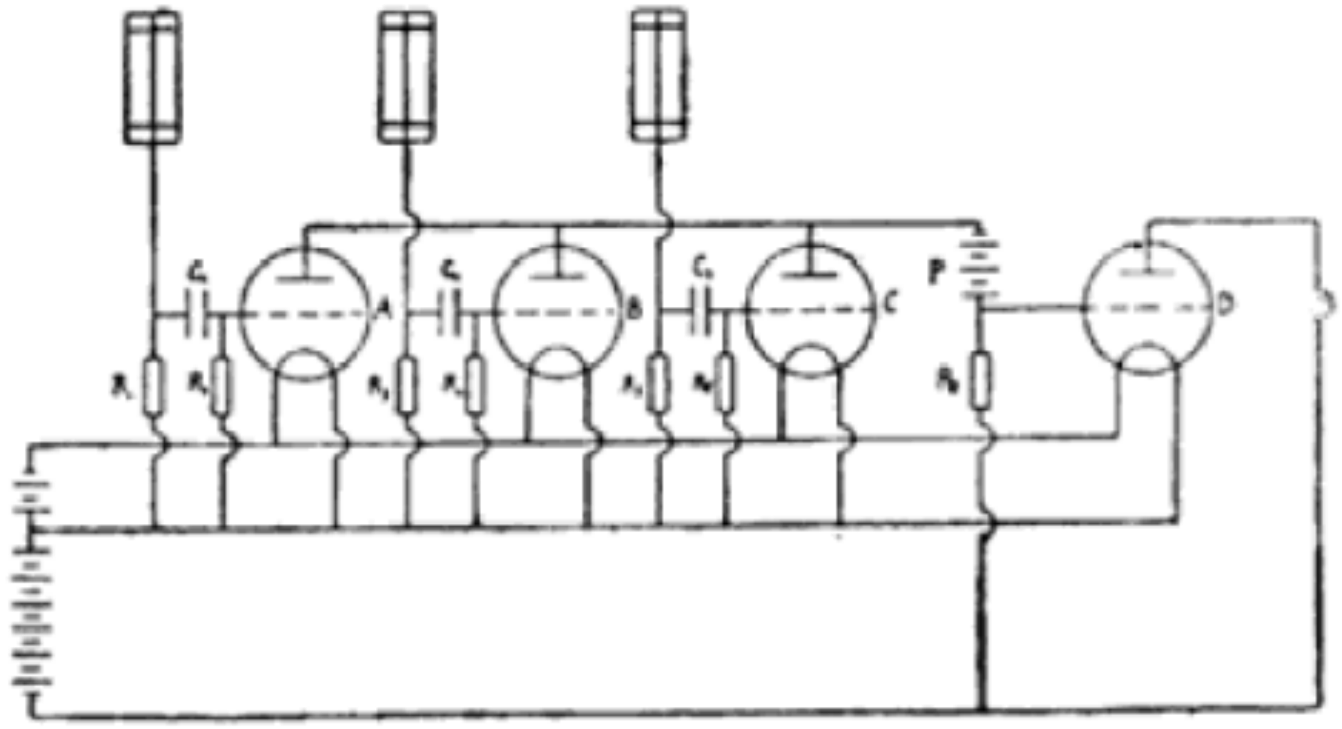
THE FIRST TRIGGER

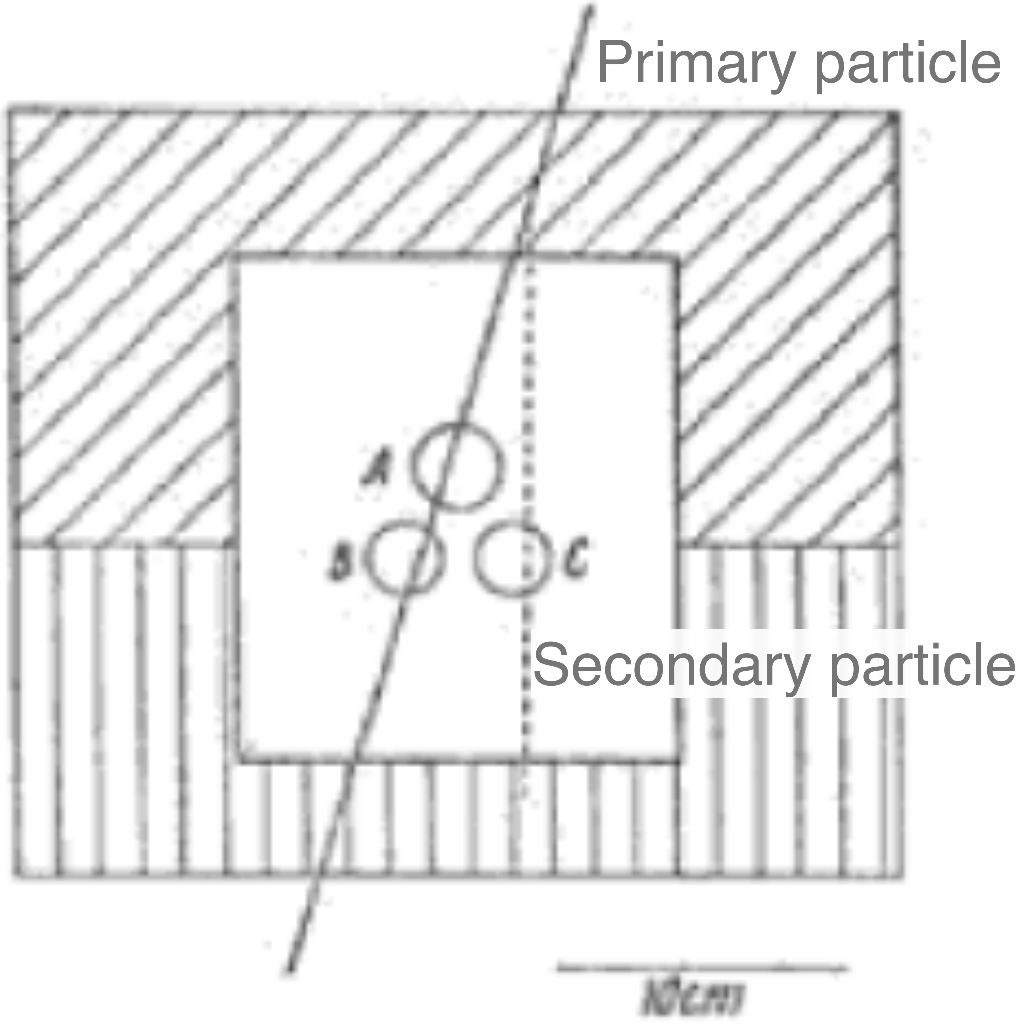
- ***"Method of Registering Multiple Simultaneous Impulses of Several Geiger Counters"***

  Bruno Rossi, Nature 1930

  – Online coincidence of three signals

Geiger counters



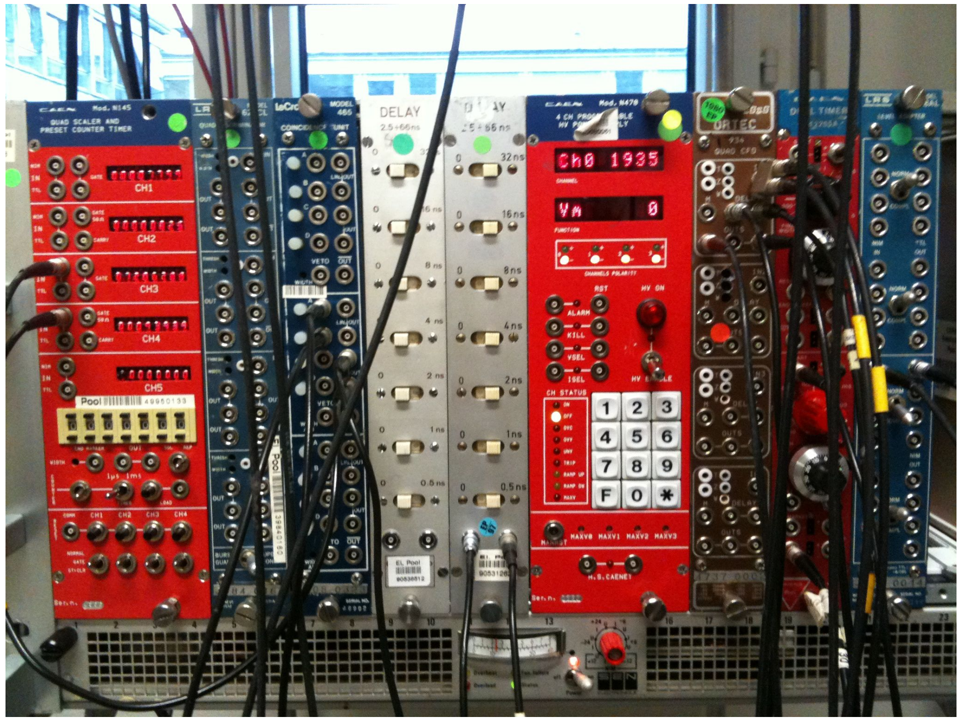

Primary particle

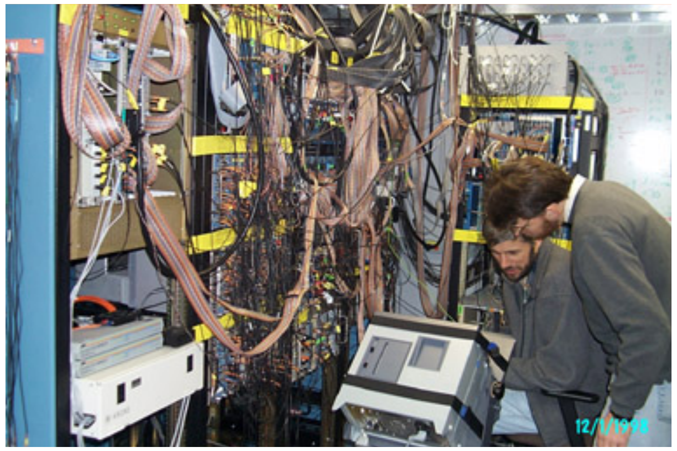
Secondary particle

Astronomia e Fisica a Firenze: dalla Specola ad Arcetri, Firenze Universiry Press, 2017

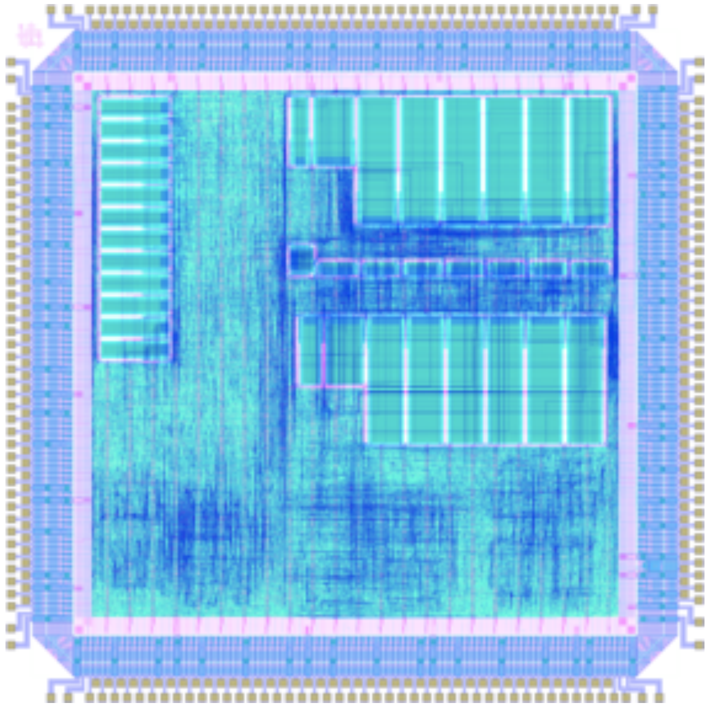
CHOOSE YOUR HARDWARE TRIGGER

- **Modular electronics**
  - Simple algorithms
  - Low-cost
  - Intuitive and fast use





- **Digital integrated systems**
  - Highly complex algorithms
  - Fast signals processing
  - Knowledge of digital systems



THREE TYPES OF TRIGGER

THREE TYPES OF TRIGGER

- **Global:**
  - an external system identifies the “interesting” event, all the readout data is collected for that **event identifier**

THREE TYPES OF TRIGGER

- ➡ **Global:**
  - ➡ an external system identifies the “interesting” event, all the readout data is collected for that **event identifier**
- ➡ **Local:**
  - ➡ local trigger decision to readout data on the **local front-end modules**, readout collects fragments corresponding to that trigger

THREE TYPES OF TRIGGER

- **Global:**
  - an external system identifies the “interesting” event, all the readout data is collected for that **event identifier**
- **Local:**
  - local trigger decision to readout data on the **local front-end modules**, readout collects fragments corresponding to that trigger
- **Continuous readout:**
  - front-end sends data continuously to the readout, at a fixed rate, regardless the data content. **Data size and rate are constant in size**. Readout cannot group fragments relative to an event

By Rick Harrison (license)

not really a photo, almost a movie

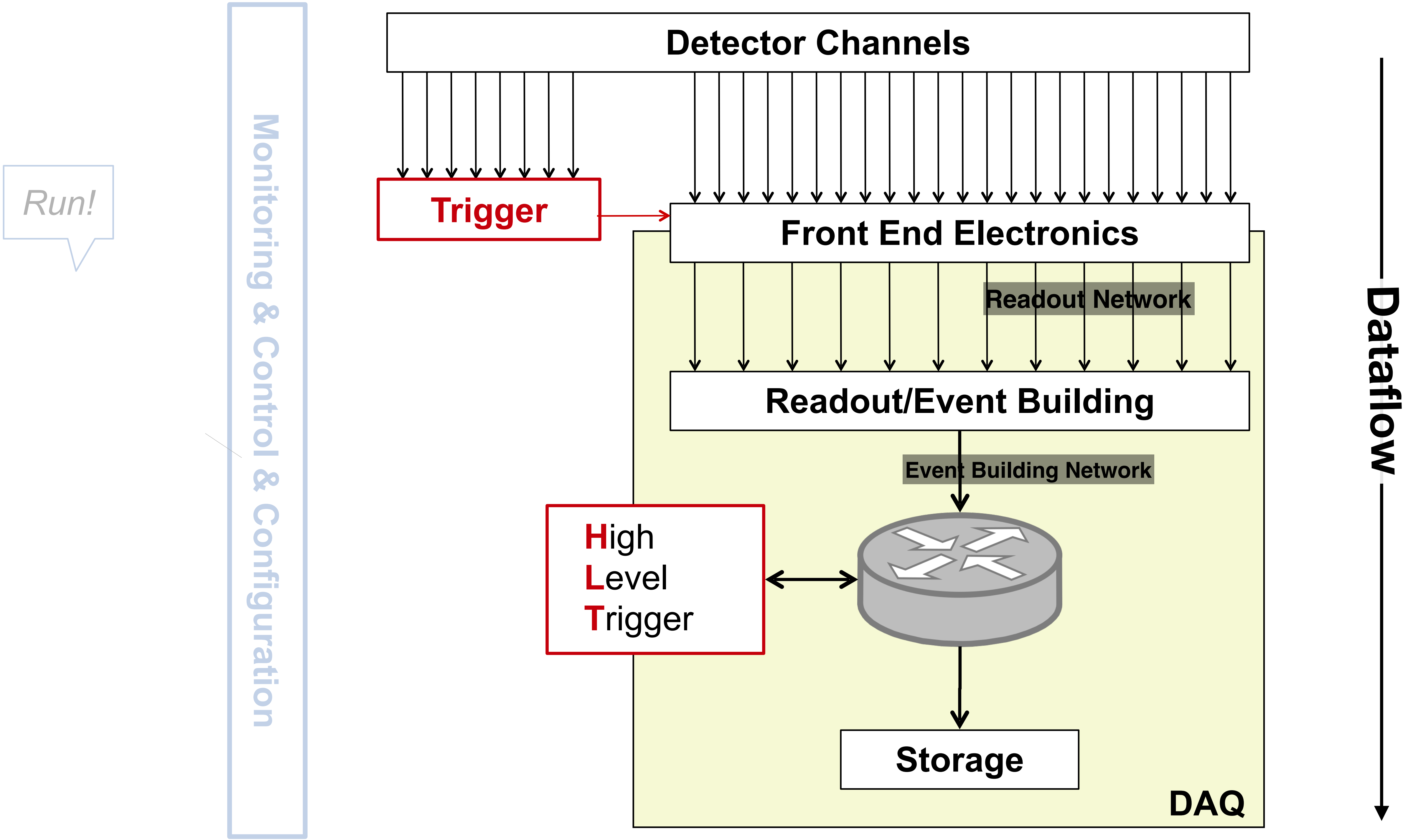
THREE TYPES OF TRIGGER

- ➡ **use cases:**
  - ➡ **Colliders**: normally use global trigger: if something interesting has been seen somewhere, take all the data corresponding to that bunch crossing
  - ➡ **Large distributed telescopes**: often use local trigger: readout data for the portions of the detector that have seen something
  - ➡ **Very slow detectors**: sometimes use continuous readout: sample the analogue signals at a fixed rate and let the downstream DAQ decide whether there were any interesting signals

By Rick Harrison (license)

not really a photo, almost a movie

T/DAQ ARCHITECTURE

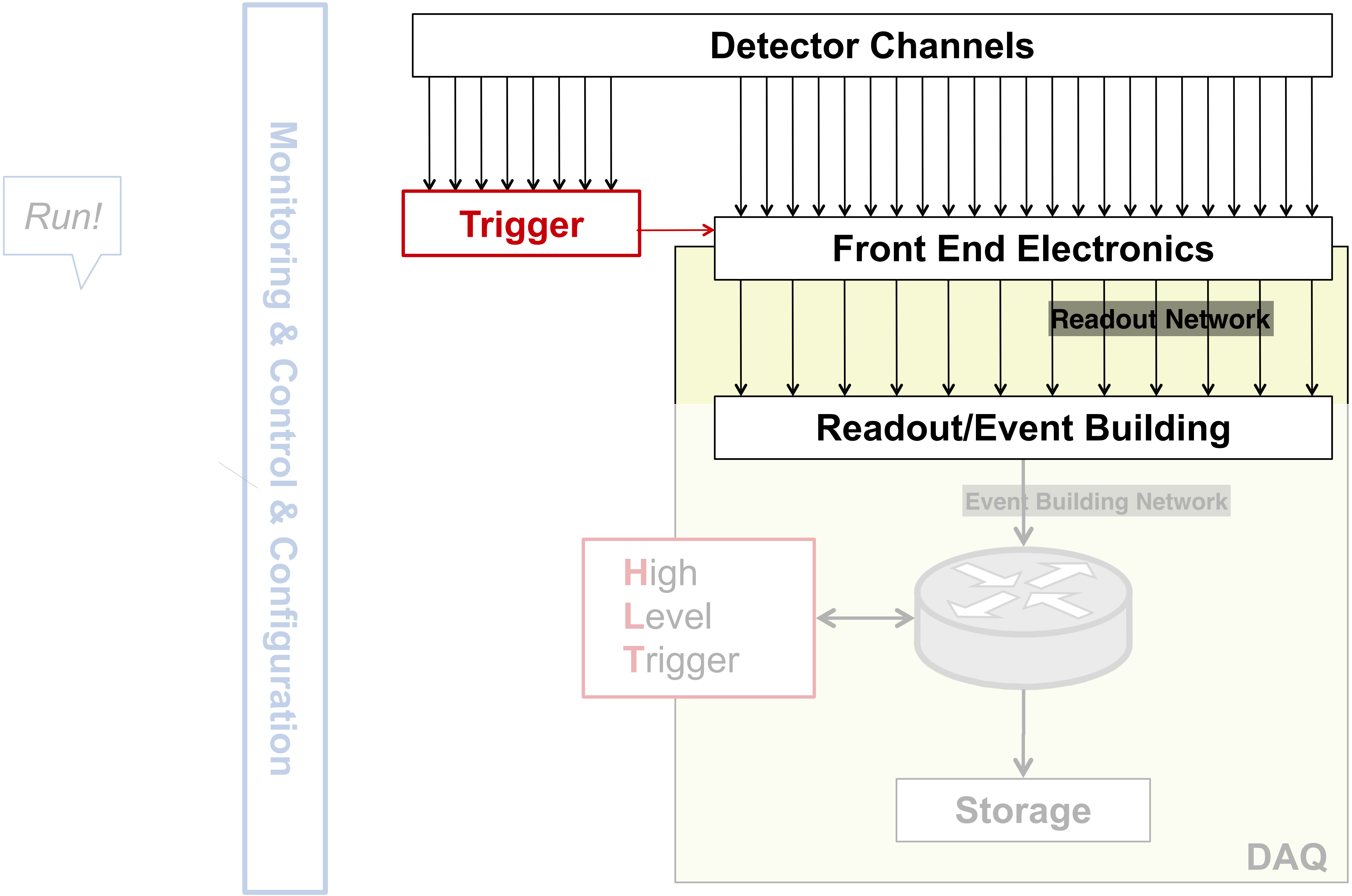


DAQ DUTIES

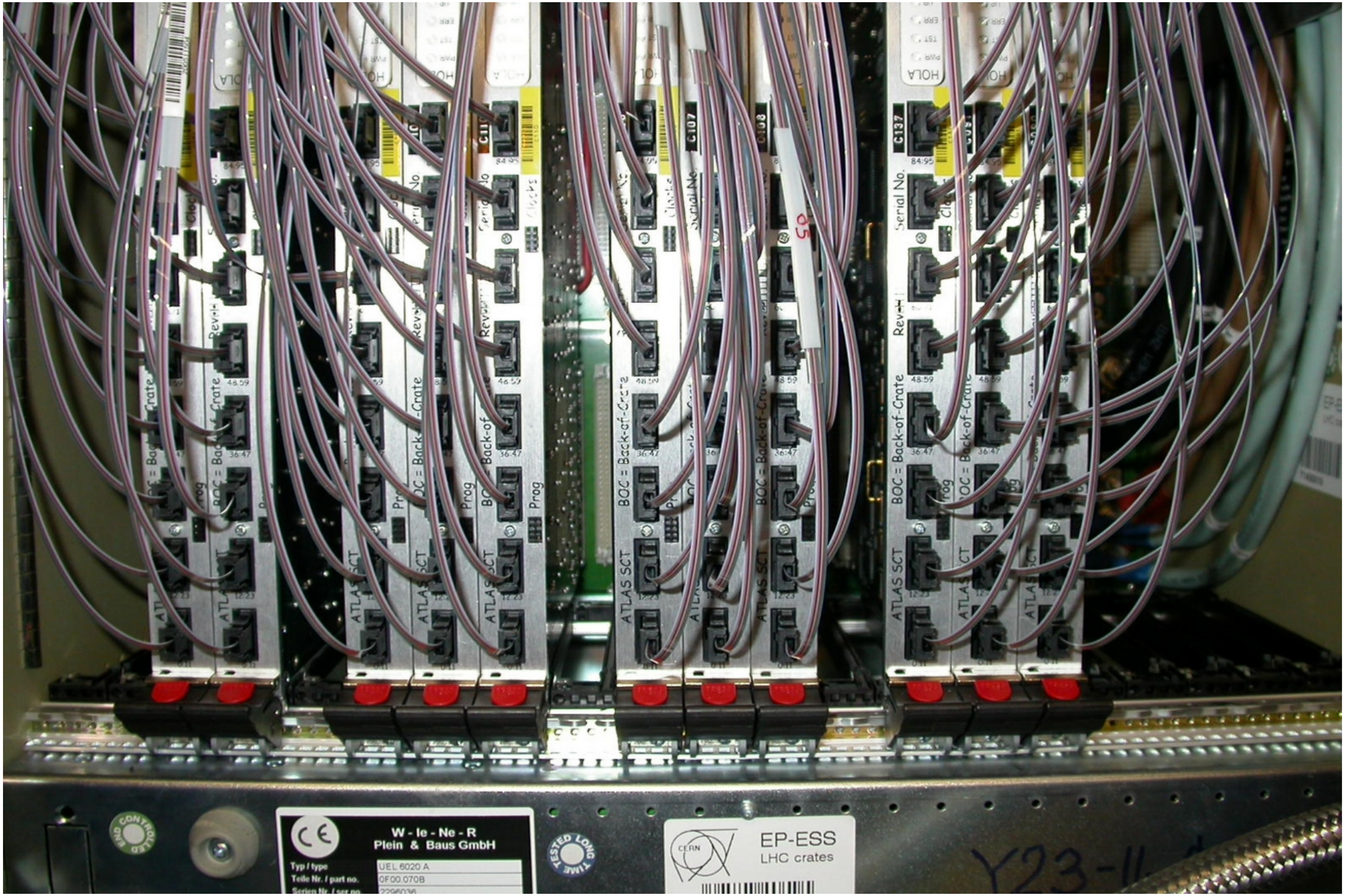
- **Gather data produced by detectors**
  - Readout
- **Form complete events**
  - Data Collection and Event Building
- **Possibly feed other trigger levels**
  - High Level Trigger
- **Store event data**
  - Data Logging
- **Manage the operations**
  - Run Control, Configuration, Monitoring

Data Flow

T/DAQ ARCHITECTURE



READOUT BOARDS (COUNTING ROOM)

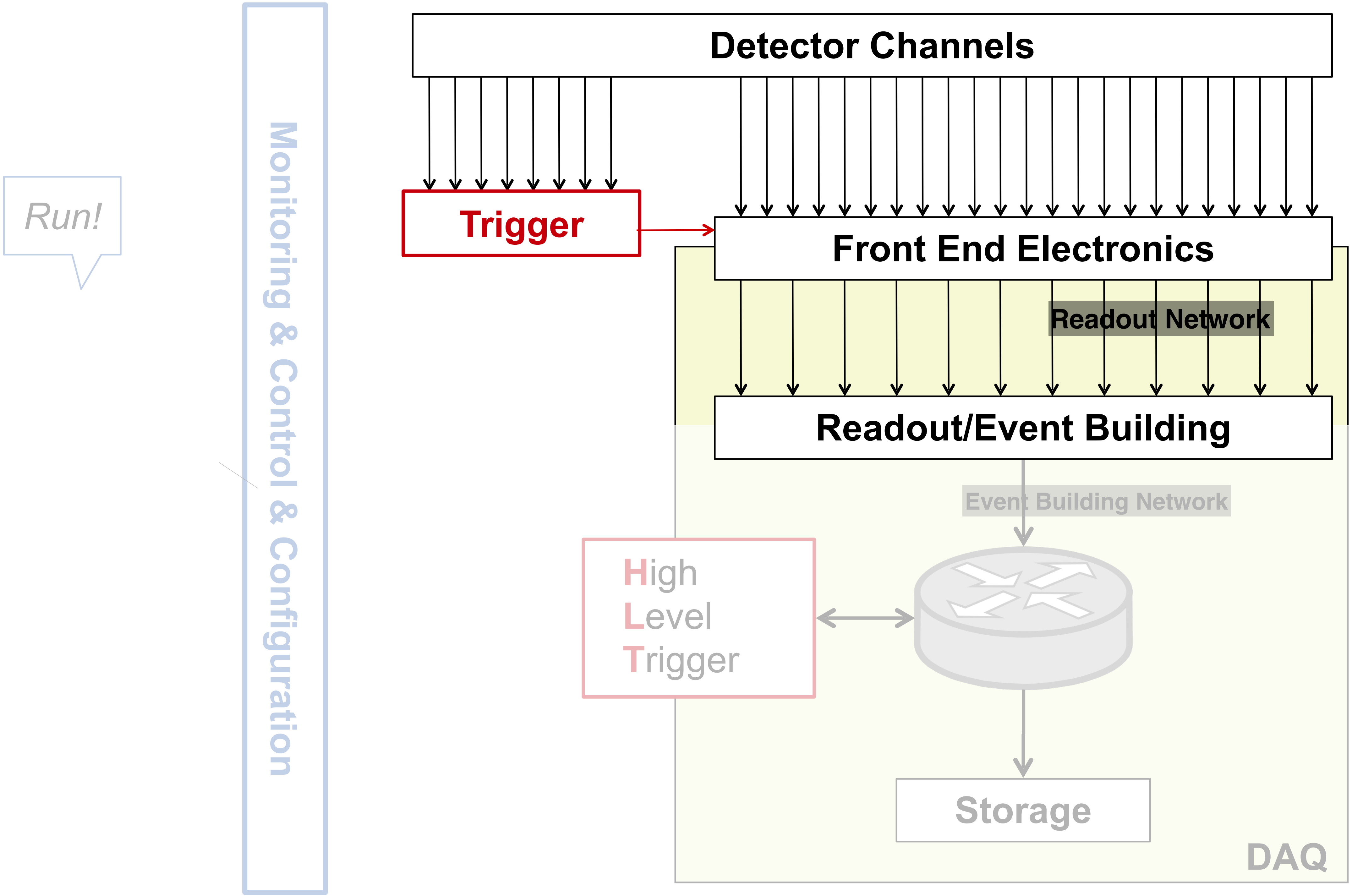


➡ **Intermediate crates off-detector to separate FE (long duration) and PCs**

READOUT

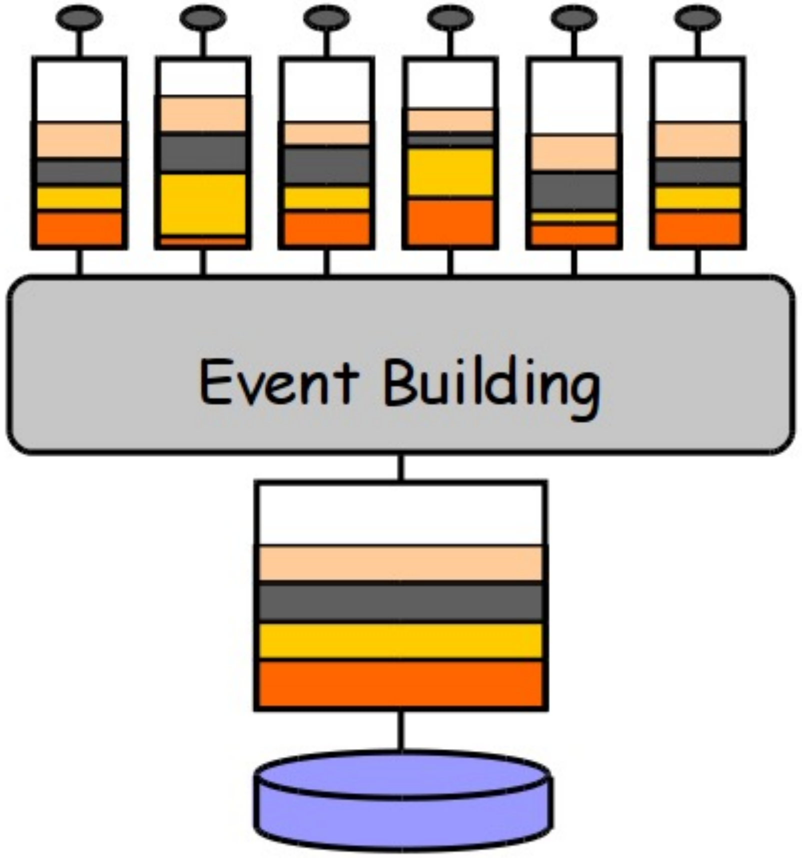
- ➡ **Signal processing, data formatting, parallelizable tasks (pattern recognition), machine learning, ...**
  - ➡ FPGAs are becoming the bread&butter of TDAQ
- ➡ **High-speed serial links, electrical and optical, depending on distance**
  - ➡ Low-power LVDS, **400 Mbps**, < 10m
  - ➡ Optical GHz-links for longer distances (up to 100 m)
- ➡ **High density backplanes for data exchanges in crates**
  - ➡ High pin count, with point-to-point connections up to **160 Mbit/s**
  - ➡ Large boards preferred

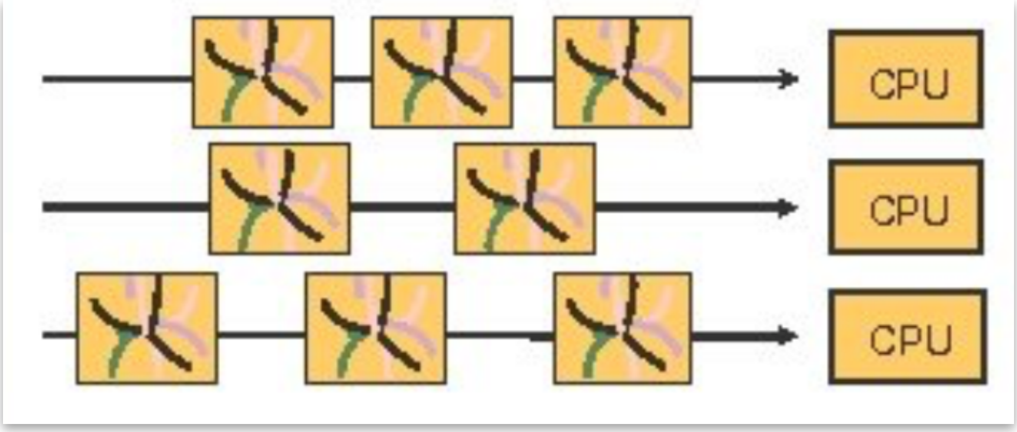
T/DAQ ARCHITECTURE



EVENT BUILDING

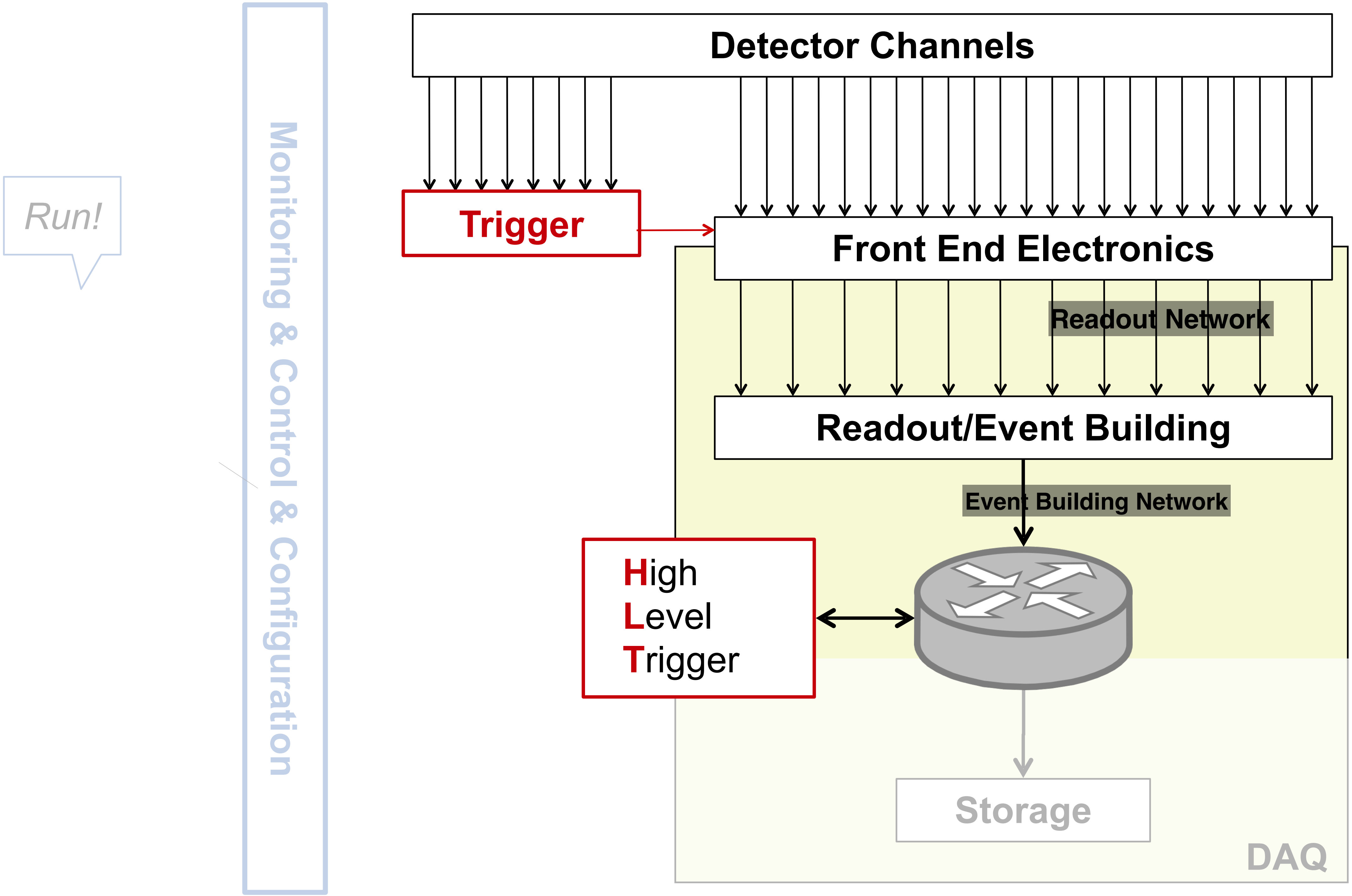
- ➡ **Collects data from Front-End and associate fragments corresponding to**
  - ➡ the same event, e.g. same bunch crossing
  - ➡ the same accelerator orbit
  - ➡ the same time frame
- ➡ **Data must be marked with time-stamp**
- ➡ **Work done with: a distribution system (networks) and processing units (switch/PCs/custom board)**
  - ➡ Can be incremental on multiple networks
  - ➡ Or a separate network for data collection
- ➡ **Usually adopt the farm architecture**
  - ➡ assign one event per processor (node)
  - ➡ larger latency, but scalable
- ➡ **Network has intrinsic latency, so traffic control is critical**
  - ➡ can have one network only for traffic control



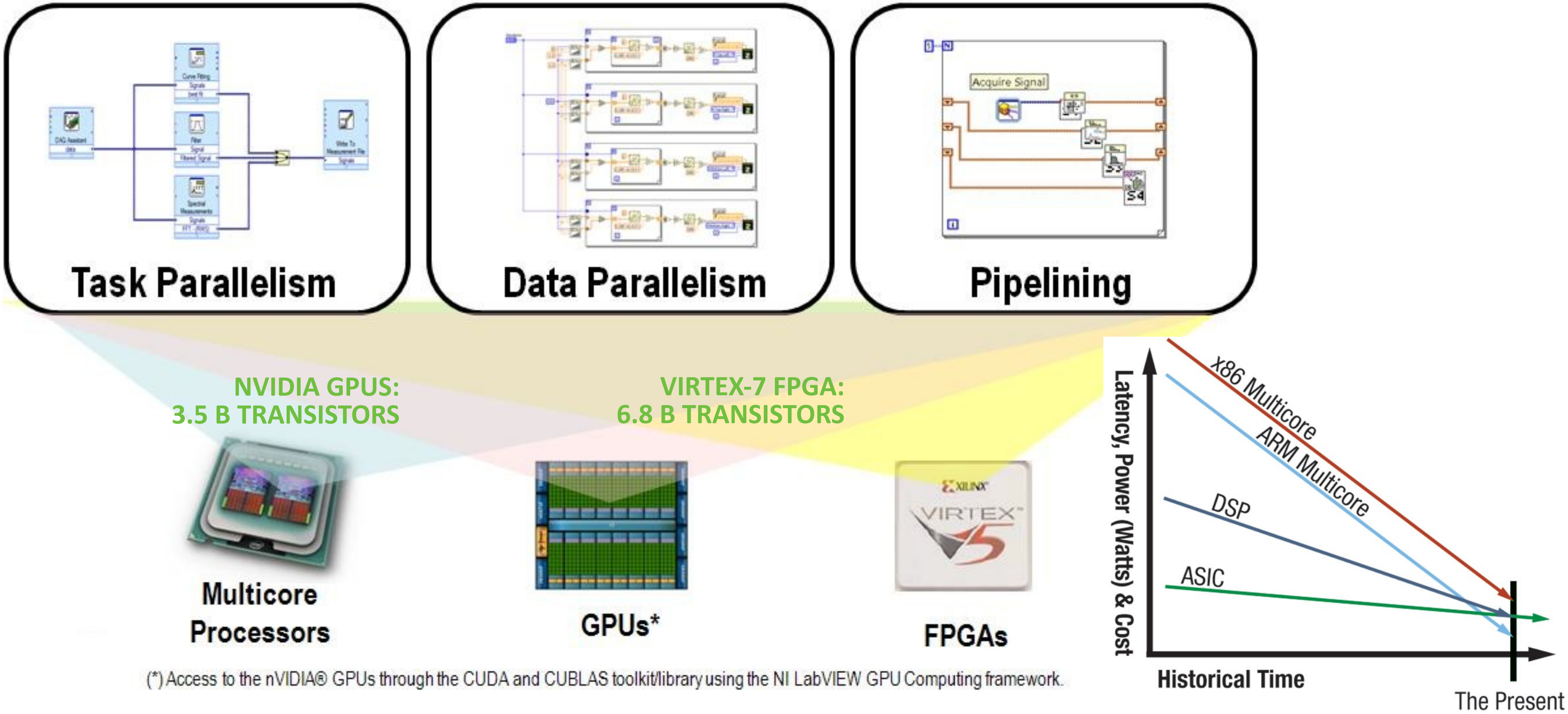


FARM (@SURFACE)

T/DAQ ARCHITECTURE



HLT TRENDS: COMBINED TECHNOLOGY



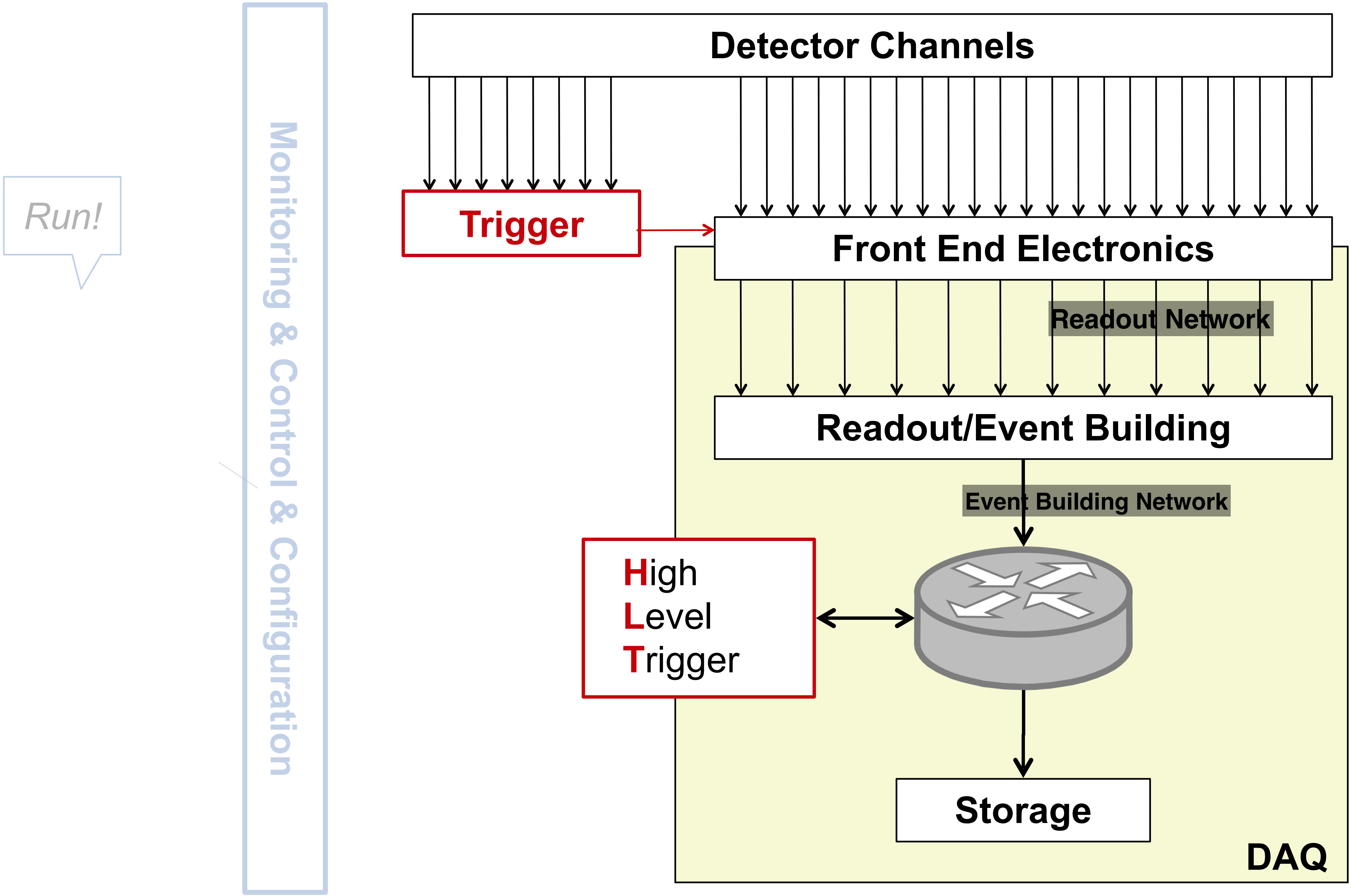
NVIDIA GPUS:
3.5 B TRANSISTORS

VIRTEX-7 FPGA:
6.8 B TRANSISTORS

(*) Access to the nVIDIA® GPUs through the CUDA and CUBLAS toolkit/library using the NI LabVIEW GPU Computing framework.

The right choice can be combining the best of both worlds by analysing which strengths of FPGA, GPU and CPU best fit the different demands of the application.

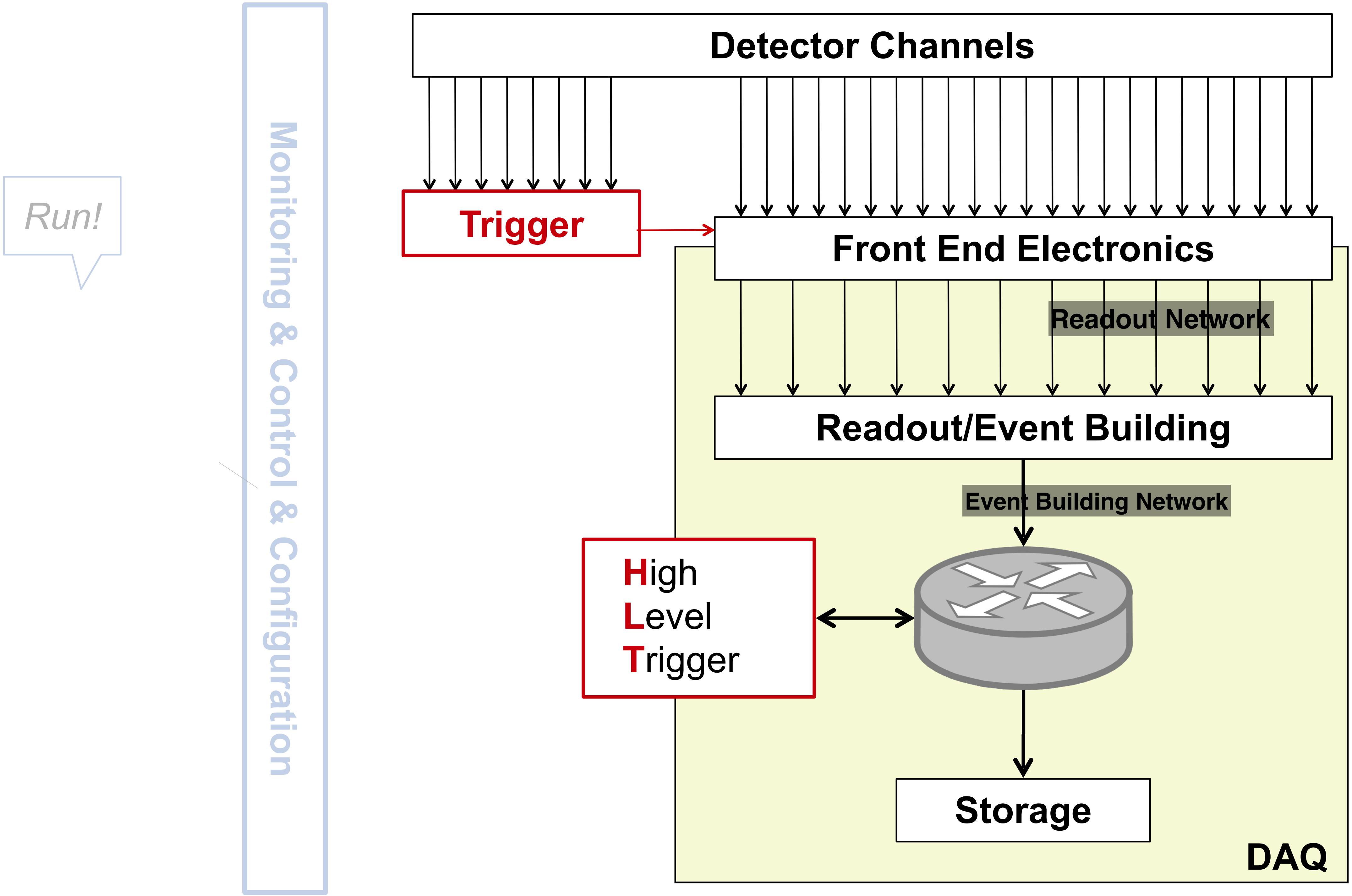
T/DAQ ARCHITECTURE



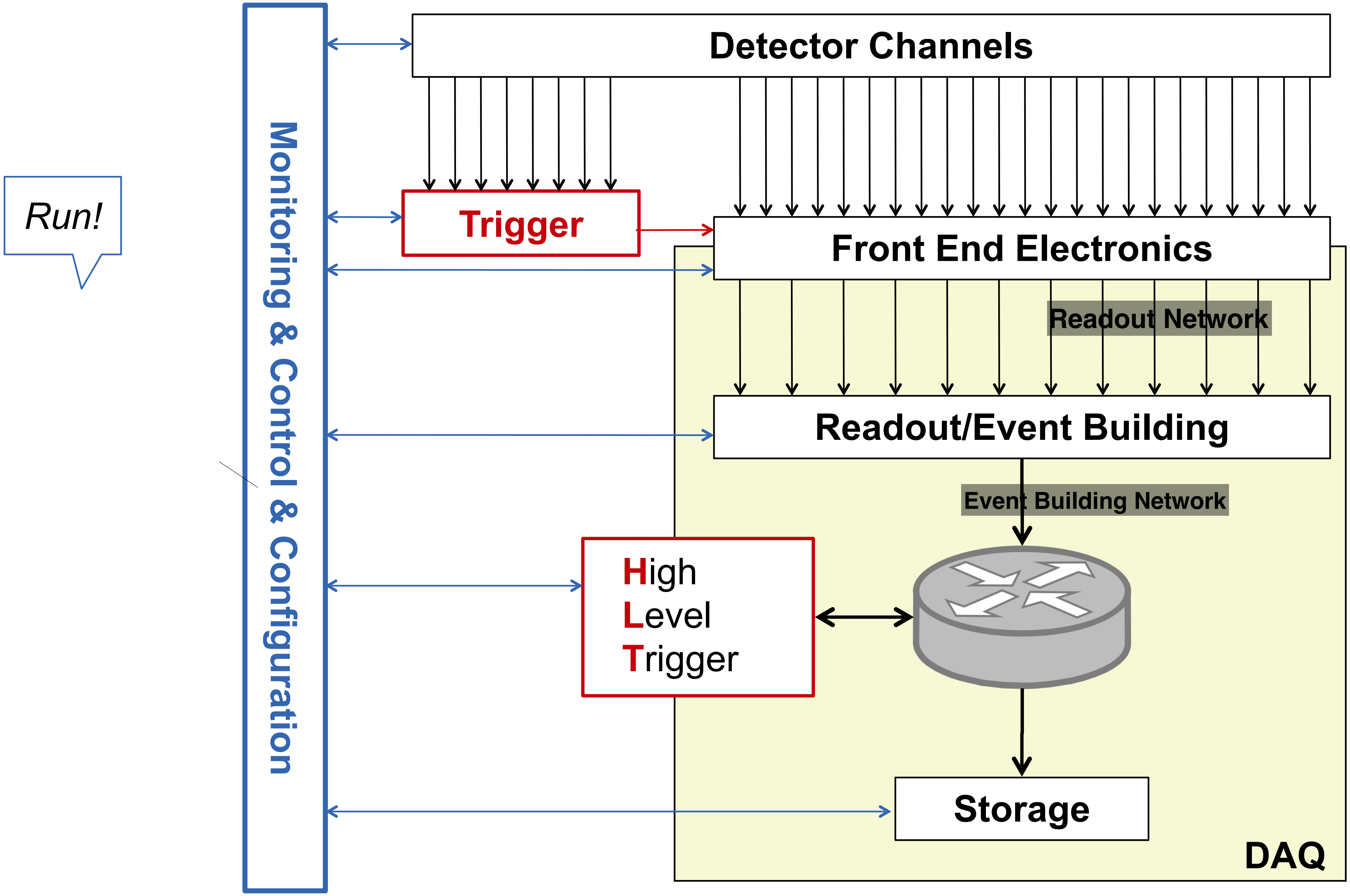
EVENT BUILDING AND STORAGE

- **Storage device technologies gaining importance in HEP**
  - Storage data rate increasing with luminosity
  - Distributed file systems being used as data-flow frameworks
    - CMS, ATLAS run 4 (?), …
  - Also use large temporary buffers with high rate access
    - LHCb: 40 PB (3000 hard-disks) enough for days
    - SSD faster but have short lifetime wrt high read-write rate, so prefer hard-disks

T/DAQ ARCHITECTURE

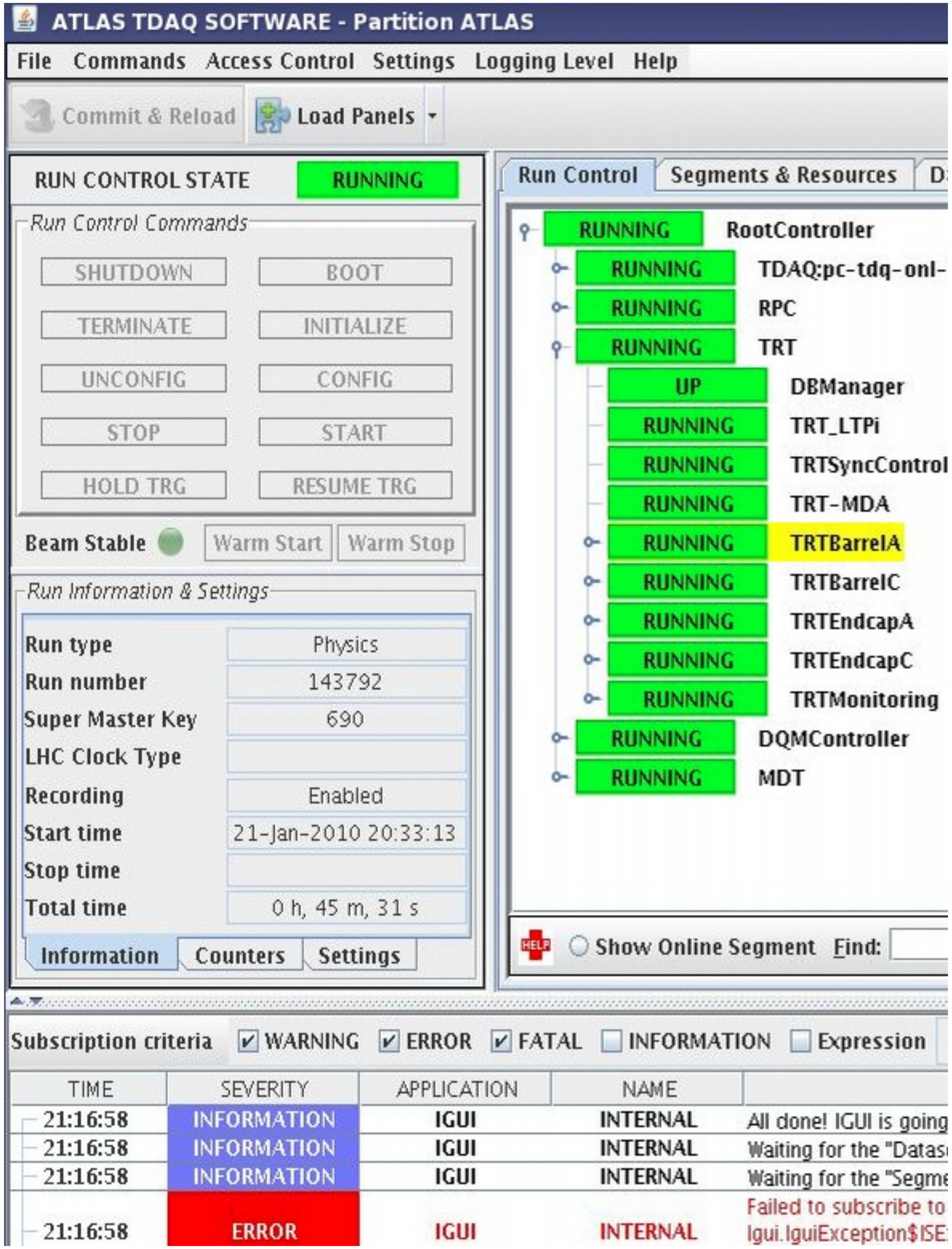


T/DAQ ARCHITECTURE



THE GLUE OF YOUR EXPERIMENT

- ➡ **Configuration**
  - ➡ data taking or test?
- ➡ **Control**
  - ➡ Orchestrate applications participating to data taking
  - ➡ Via distributed **Finite State Machine**
- ➡ **Monitoring**
  - ➡ What is going on?
  - ➡ What happened?
  - ➡ When? Where?



RUN CONTROL STATE RUNNING

Beam Stable ●

Run Information & Settings

Run type	Physics
Run number	143792
Super Master Key	690
LHC Clock Type	
Recording	Enabled
Start time	21-Jan-2010 20:33:13
Stop time	
Total time	0 h, 45 m, 31 s

RUNNING	RootController
RUNNING	TDAQ:pc-tdq-onl-
RUNNING	RPC
RUNNING	TRT
UP	DBManager
RUNNING	TRT_LTPi
RUNNING	TRTSyncControl
RUNNING	TRT-MDA
RUNNING	TRTBarrelA
RUNNING	TRTBarrelC
RUNNING	TRTEndcapA
RUNNING	TRTEndcapC
RUNNING	TRTMonitoring
RUNNING	DQMController
RUNNING	MDT

Subscription criteria ☑ WARNING ☑ ERROR ☑ FATAL □ INFORMATION □ Expression

TIME	SEVERITY	APPLICATION	NAME	
21:16:58	INFORMATION	IGUI	INTERNAL	All done! IGUI is going
21:16:58	INFORMATION	IGUI	INTERNAL	Waiting for the "Datas
21:16:58	INFORMATION	IGUI	INTERNAL	Waiting for the "Segme
21:16:58	ERROR	IGUI	INTERNAL	Failed to subscribe to Igui.IguiException$ISE

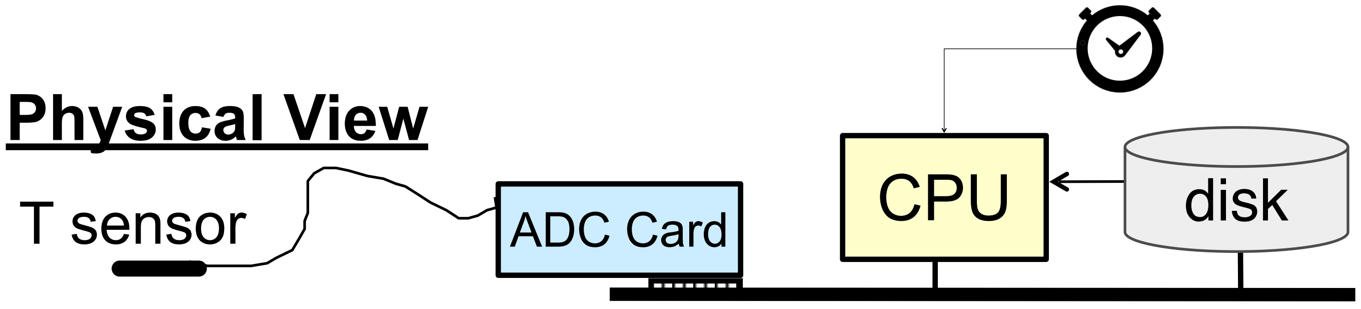
OUTLINE

- ➡ **Introduction**
    - ➡ What is Trigger and DAQ?
    - ➡ Overall TDAQ framework
- ➡ **Basic TDAQ concepts**
    - ➡ Digitization, Latency
    - ➡ Deadtime, Busy
    - ➡ De-randomization
- ➡ **Scaling up**
    - ➡ Readout and Event Building
    - ➡ Buses vs Network

Via a toy model

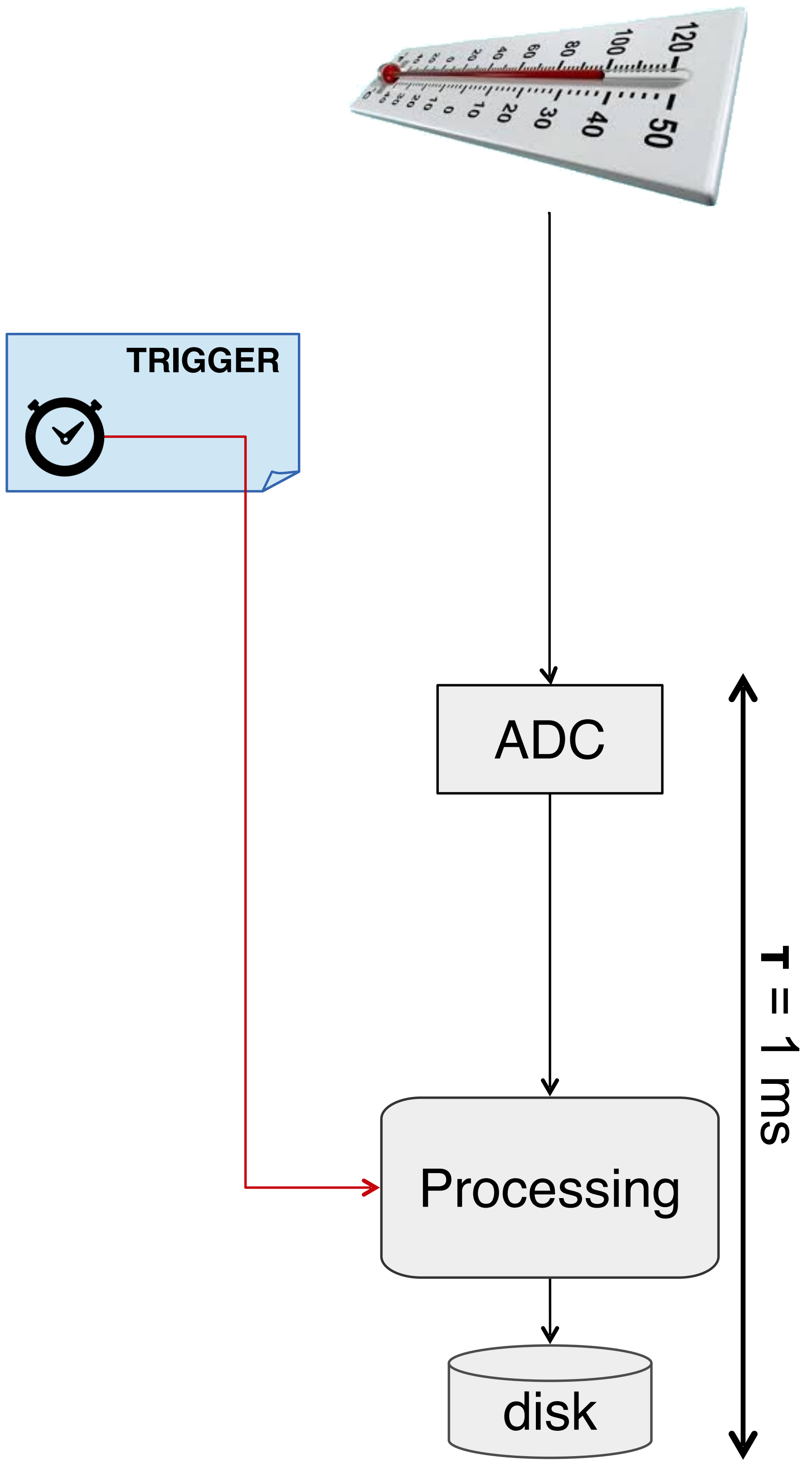
BASIC DAQ: PERIODIC TRIGGER

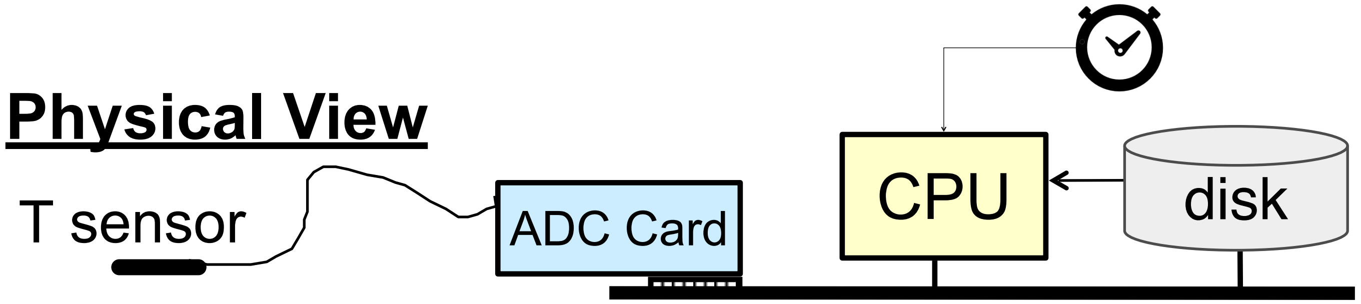
- **Eg: measure temperature at a fixed frequency**
  - Clock trigger
- **ADC performs analog to digital conversion, digitization (on front-end electronics)**
  - Encoding analog value into binary representation
- **CPU does**
  - Readout, Processing, Storage



BASIC DAQ: PERIODIC TRIGGER

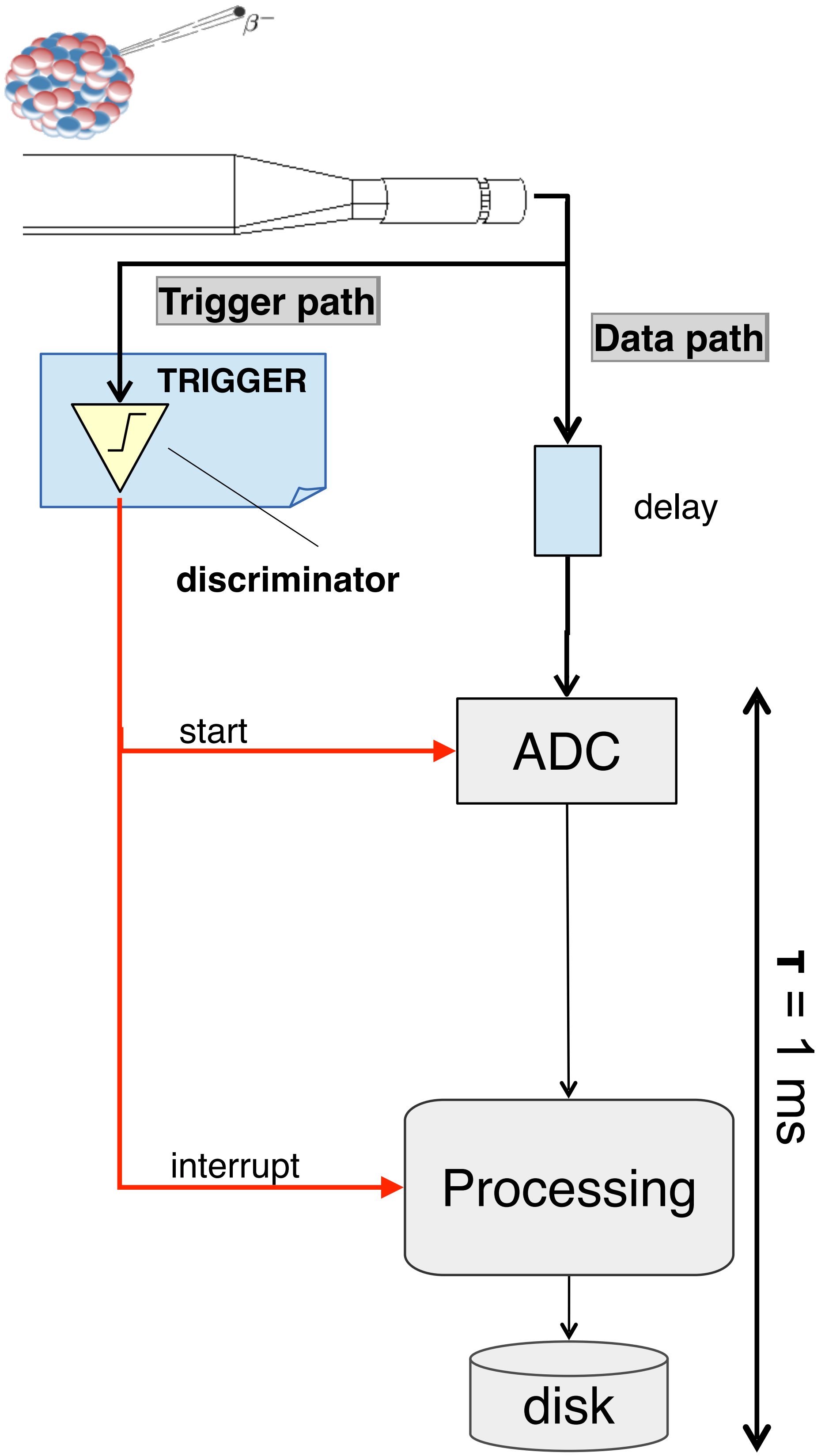
- **System clearly limited by the time *τ* to process an “event”**
  - ADC conversion + CPU processing + Storage
- **The DAQ maximum sustainable rate is simply the inverse of *τ*, e.g.:**
  - E.g.: τ = 1 ms ® R = 1/τ = 1 kHz





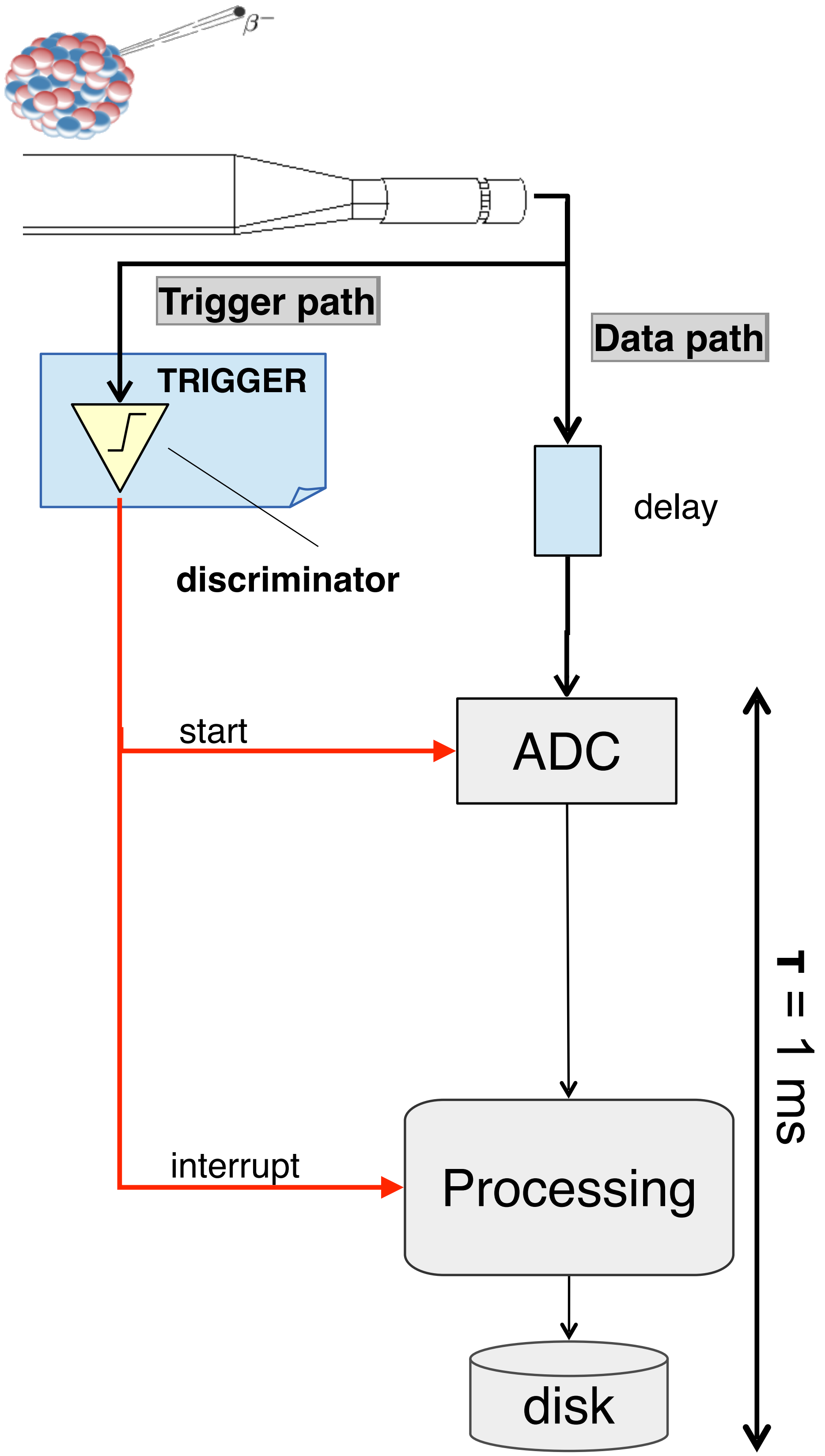
BASIC DAQ: “REAL” TRIGGER

- ➡ **Events are asynchronous and unpredictable**
  - ➡ E.g.: beta decay studies
- ➡ **A physics trigger is needed**
  - ➡ **Discriminator**: generates an output digital signal if amplitude of the input pulse is greater than a given threshold
- ➡ **NB: delay introduced to compensate for the trigger latency**
  - ➡ Signal split in trigger and data paths



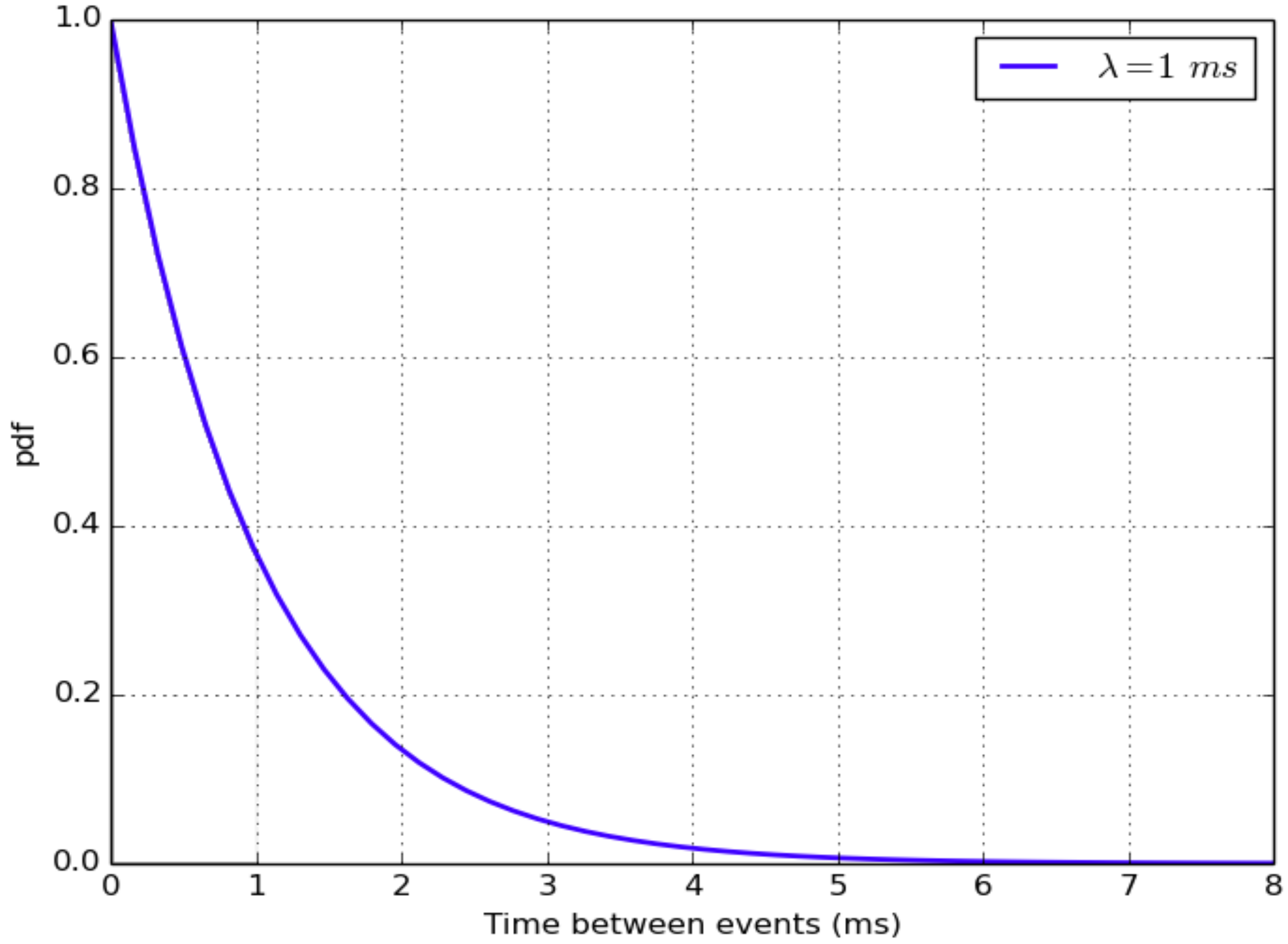
BASIC DAQ: “REAL” TRIGGER

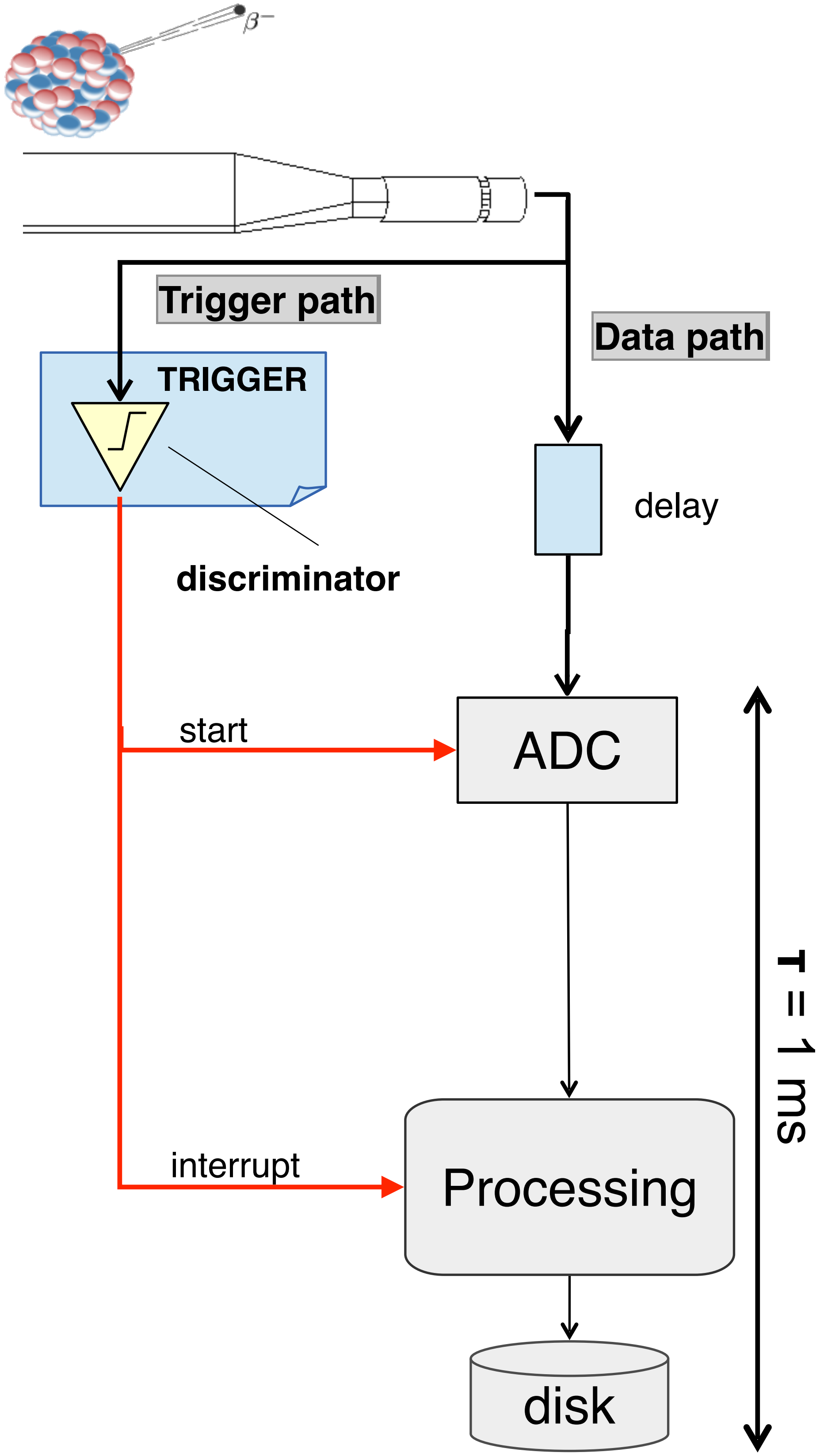
- ➡ **Events are asynchronous and unpredictable**
  - ➡ E.g.: beta decay studies
- ➡ **A physics trigger is needed**
  - ➡ **Discriminator**: generates an output digital signal if amplitude of the input pulse is greater than a given threshold
- ➡ **NB: delay introduced to compensate for the trigger latency**
  - ➡ Signal split in trigger and data paths



BASIC DAQ: “REAL” TRIGGER

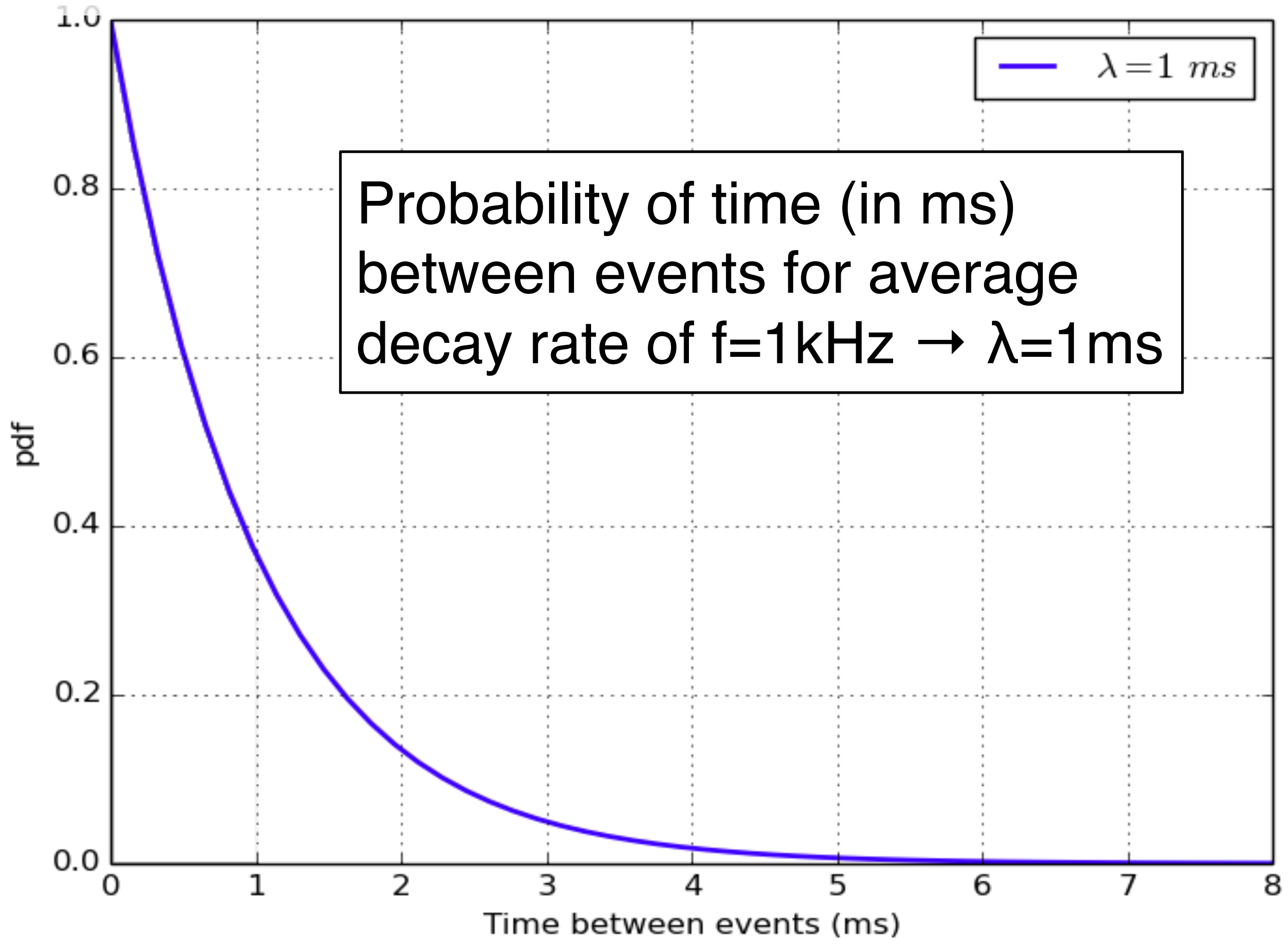
- ➡ **Stochastic process**
  - ➡ Fluctuations in time between events
- ➡ **Let's assume for example**
  - ➡ physics rate f = 1 kHz, i.e. λ = 1 ms
  - ➡ and, as before, τ= 1 ms



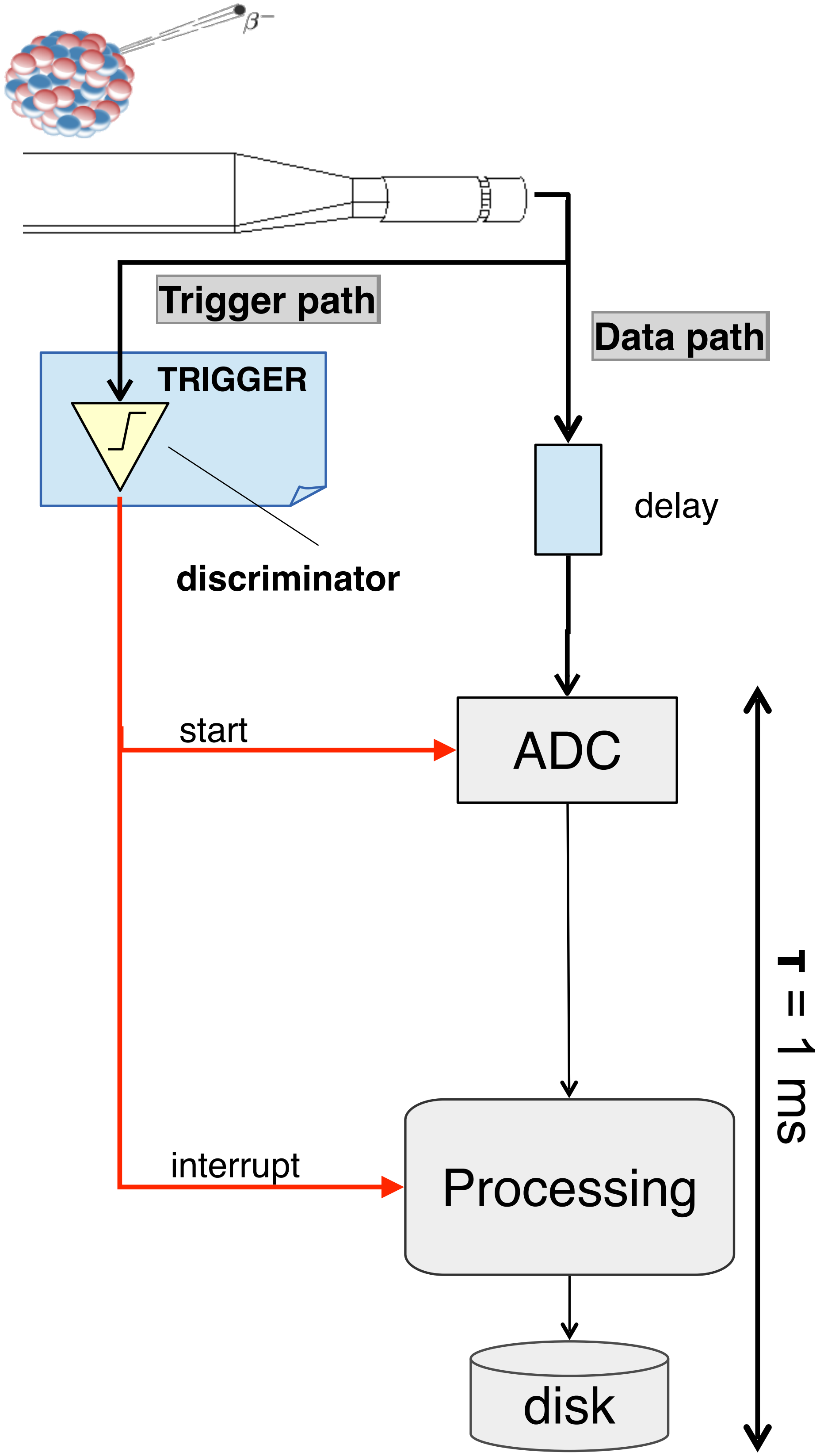


BASIC DAQ: “REAL” TRIGGER

- ➡ **Stochastic process**
  - ➡ Fluctuations in time between events
- ➡ **Let's assume for example**
  - ➡ physics rate f = 1 kHz, i.e. λ = 1 ms
  - ➡ and, as before, τ= 1 ms

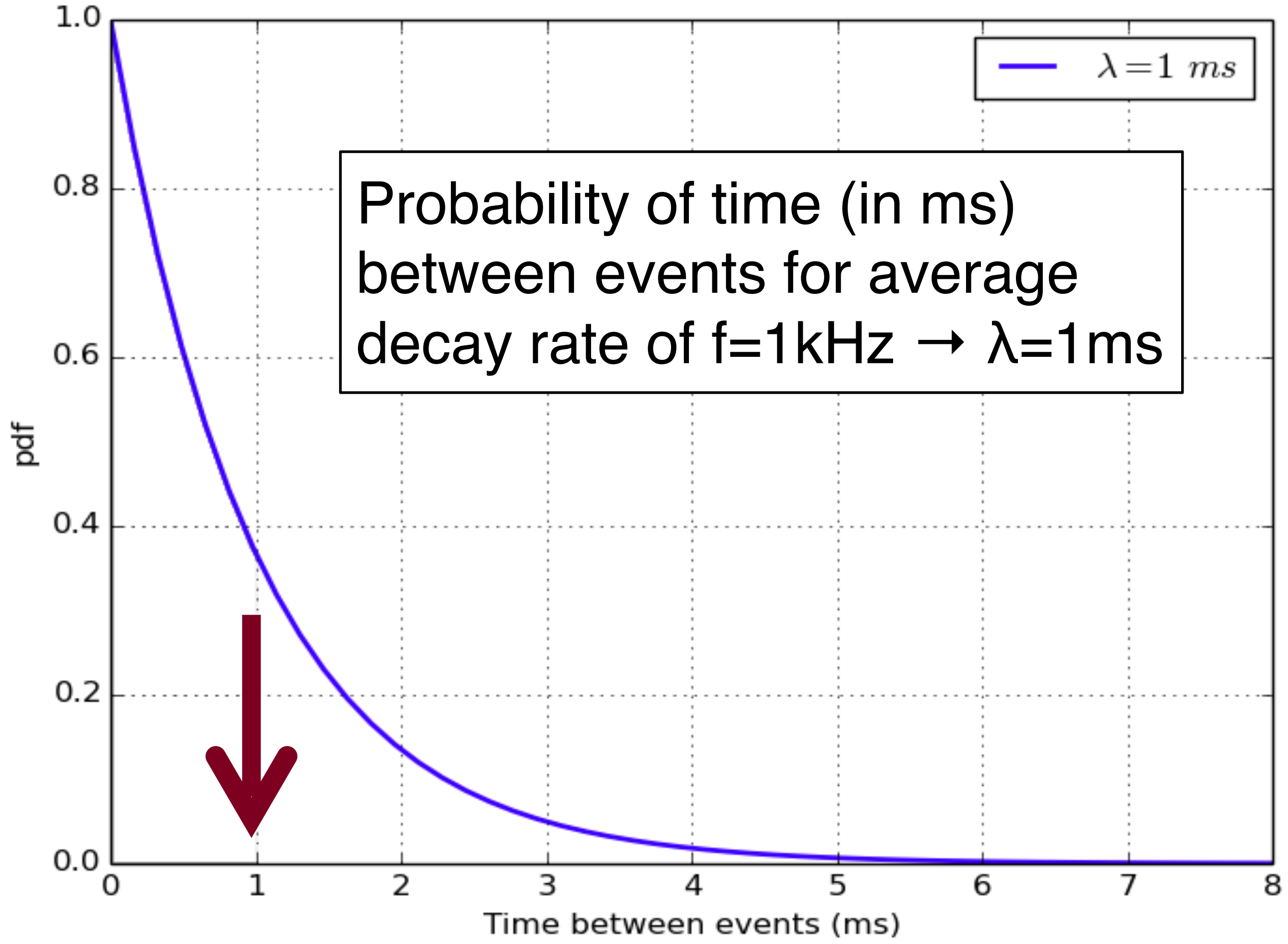


Probability of time (in ms) between events for average decay rate of f=1kHz → λ=1ms

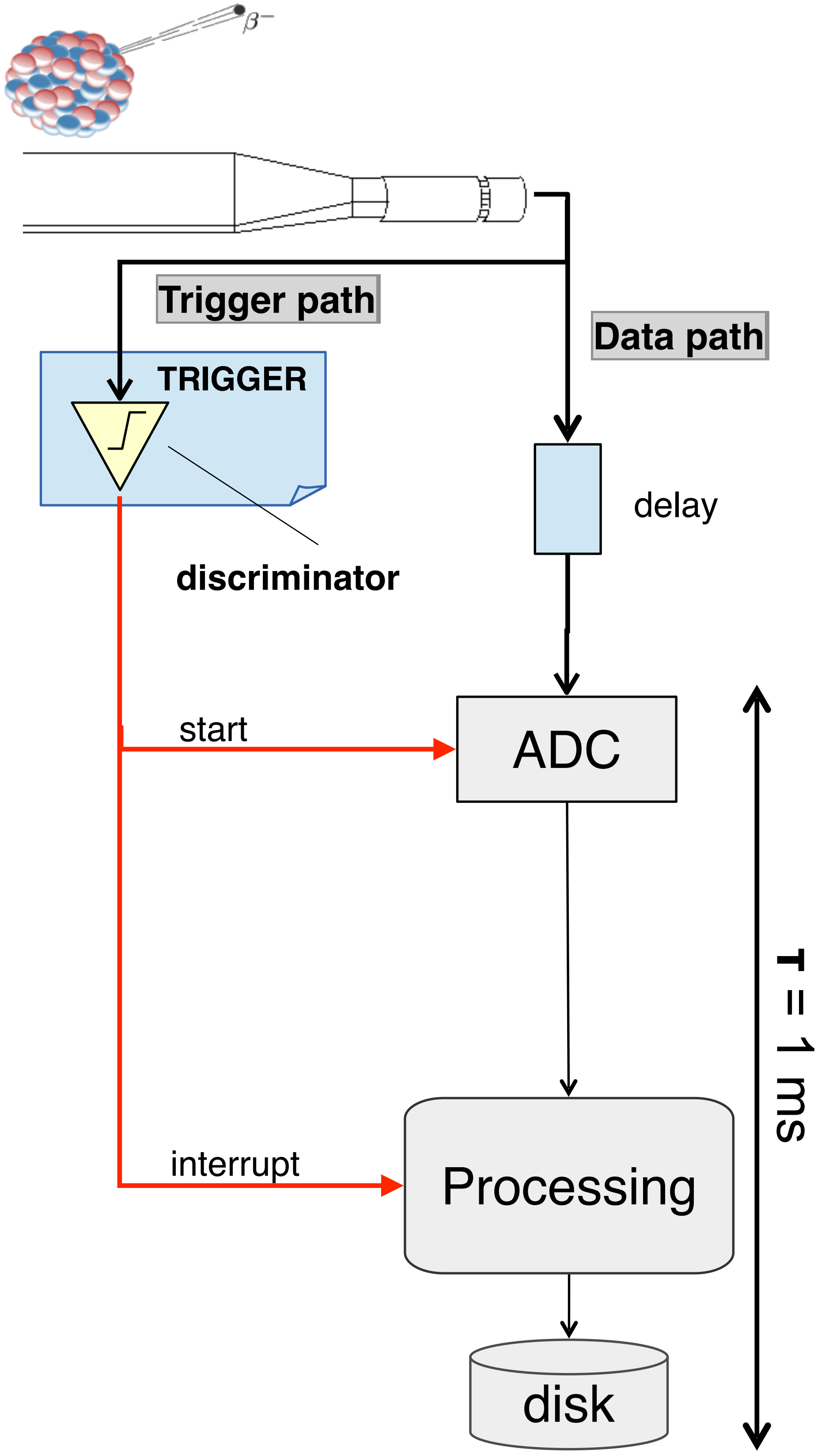


BASIC DAQ: “REAL” TRIGGER

- **Stochastic process**
  - Fluctuations in time between events
- **Let's assume for example**
  - physics rate f = 1 kHz, i.e. λ = 1 ms
  - and, as before, τ= 1 ms

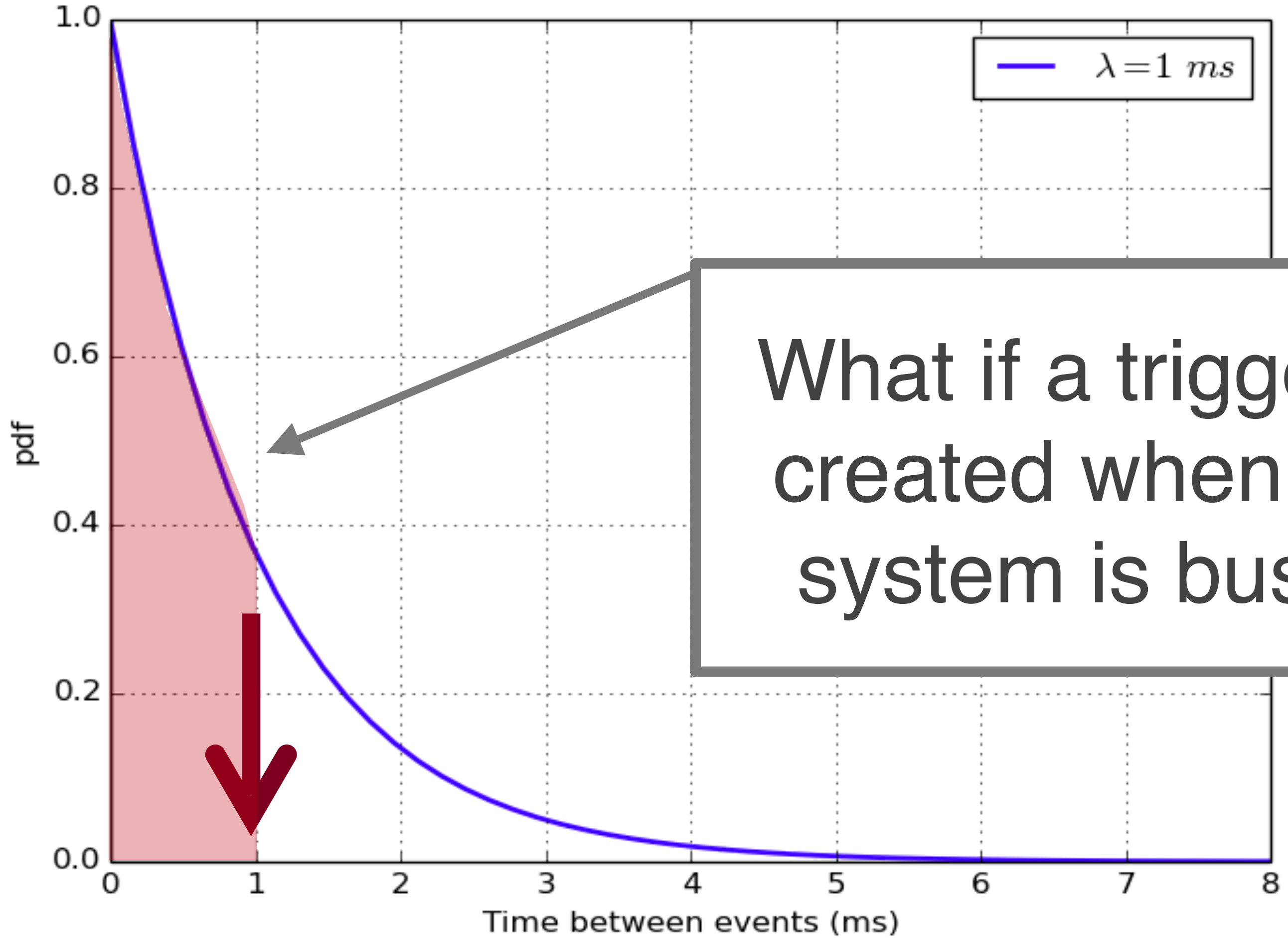


Probability of time (in ms) between events for average decay rate of f=1kHz → λ=1ms

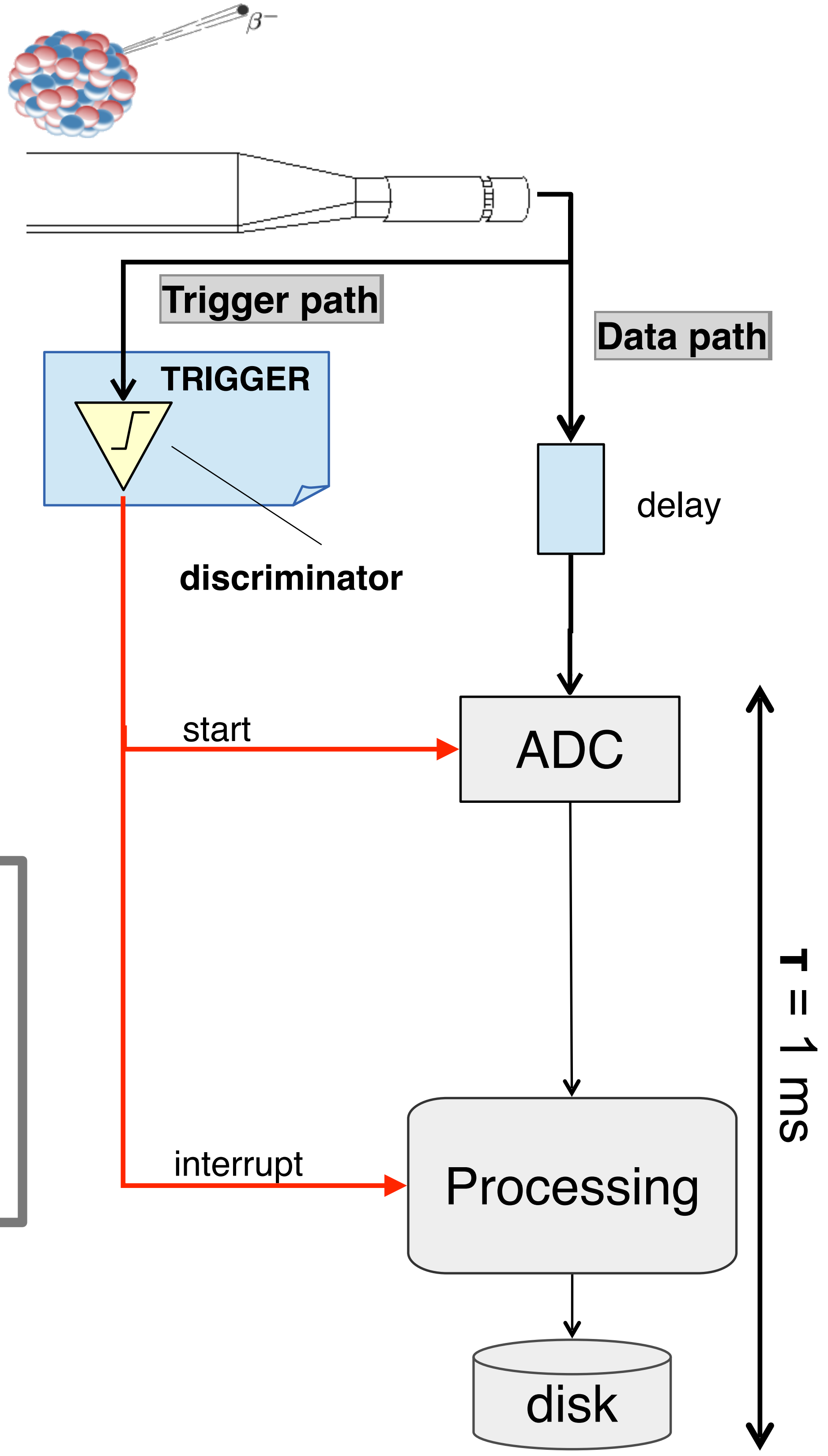


BASIC DAQ: “REAL” TRIGGER

- **Stochastic process**
  - Fluctuations in time between events
- **Let's assume for example**
  - physics rate f = 1 kHz, i.e. λ = 1 ms
  - and, as before, τ= 1 ms

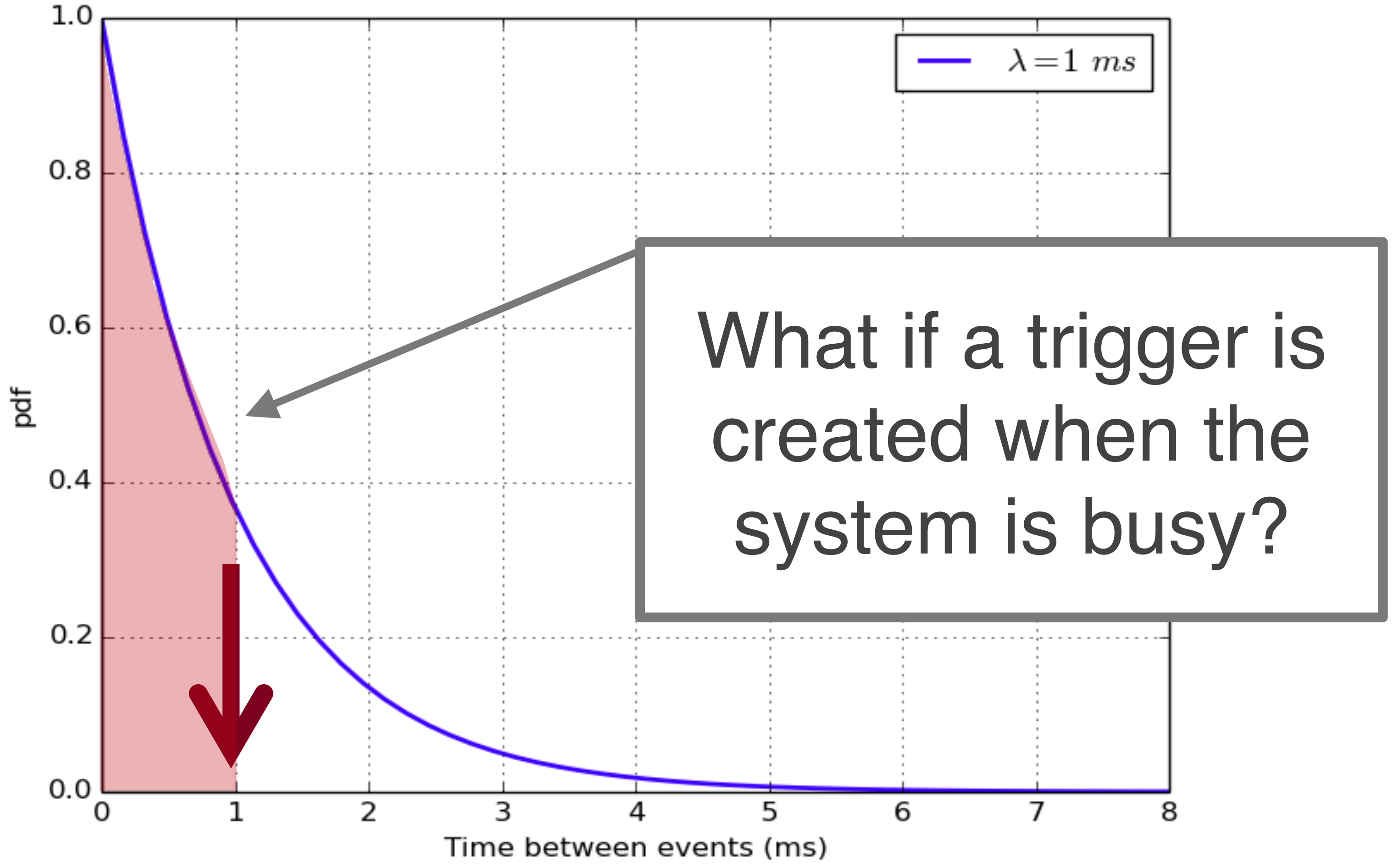


What if a trigger is created when the system is busy?

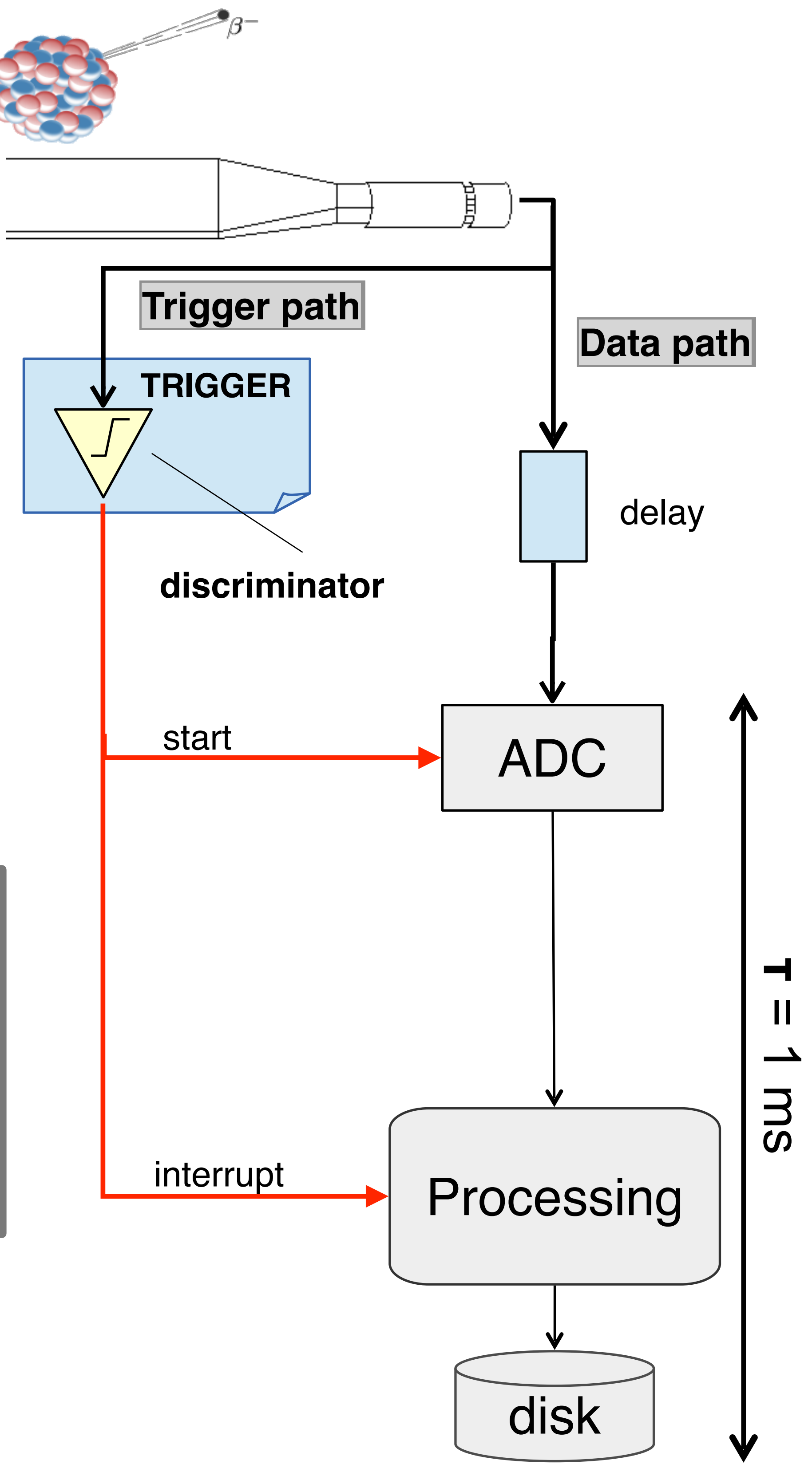


SYSTEM STILL PROCESSING

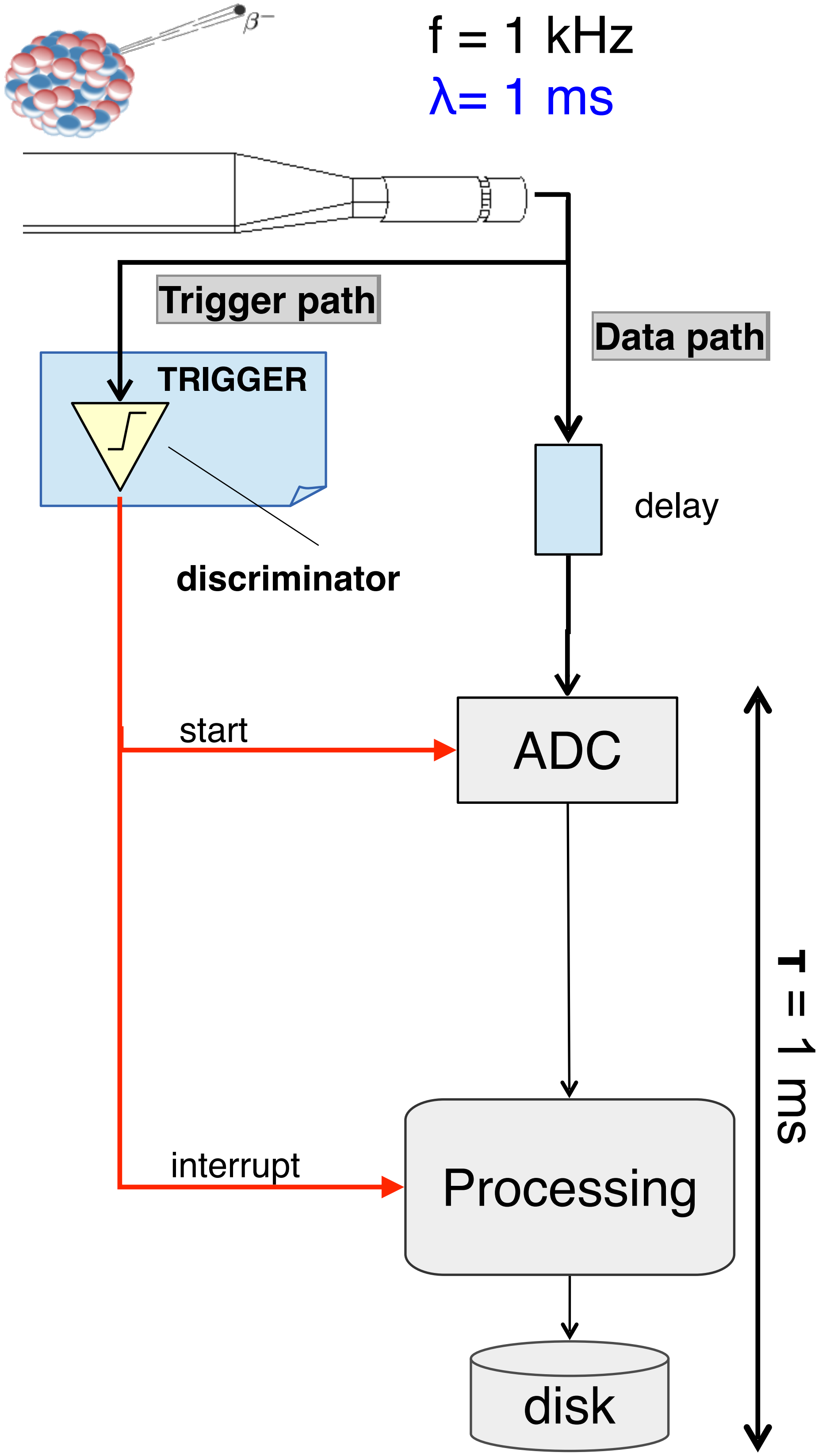
- ➡ **If a new trigger arrives when the system is still processing the previous event**
    - ➡ The processing of the previous event could be screwed up



What if a trigger is created when the system is busy?

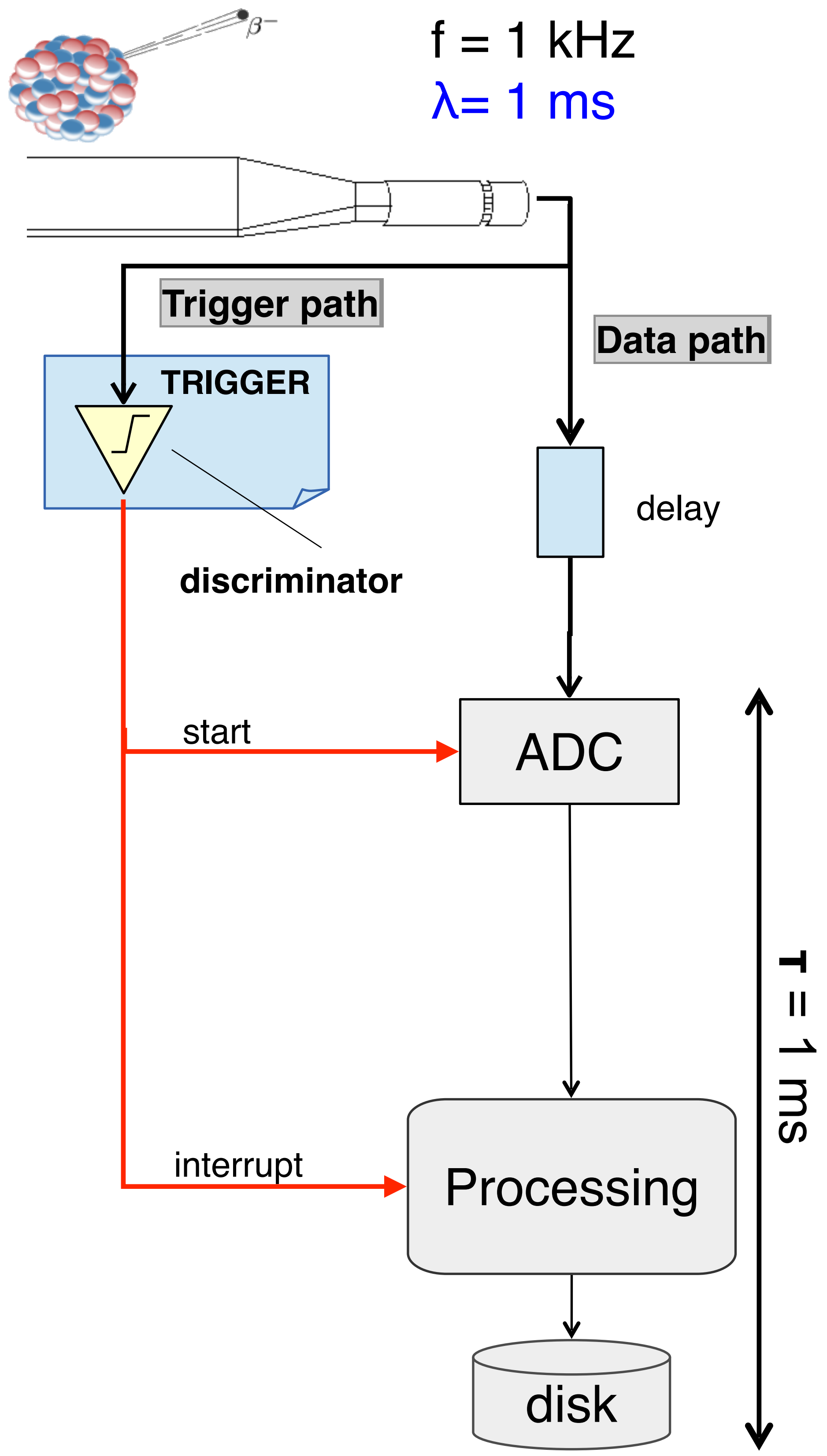


BUSY LOGIC


β⁻
f = 1 kHz
λ= 1 ms
Trigger path
Data path
TRIGGER
discriminator
delay
start
ADC
interrupt
Processing
disk
τ = 1 ms

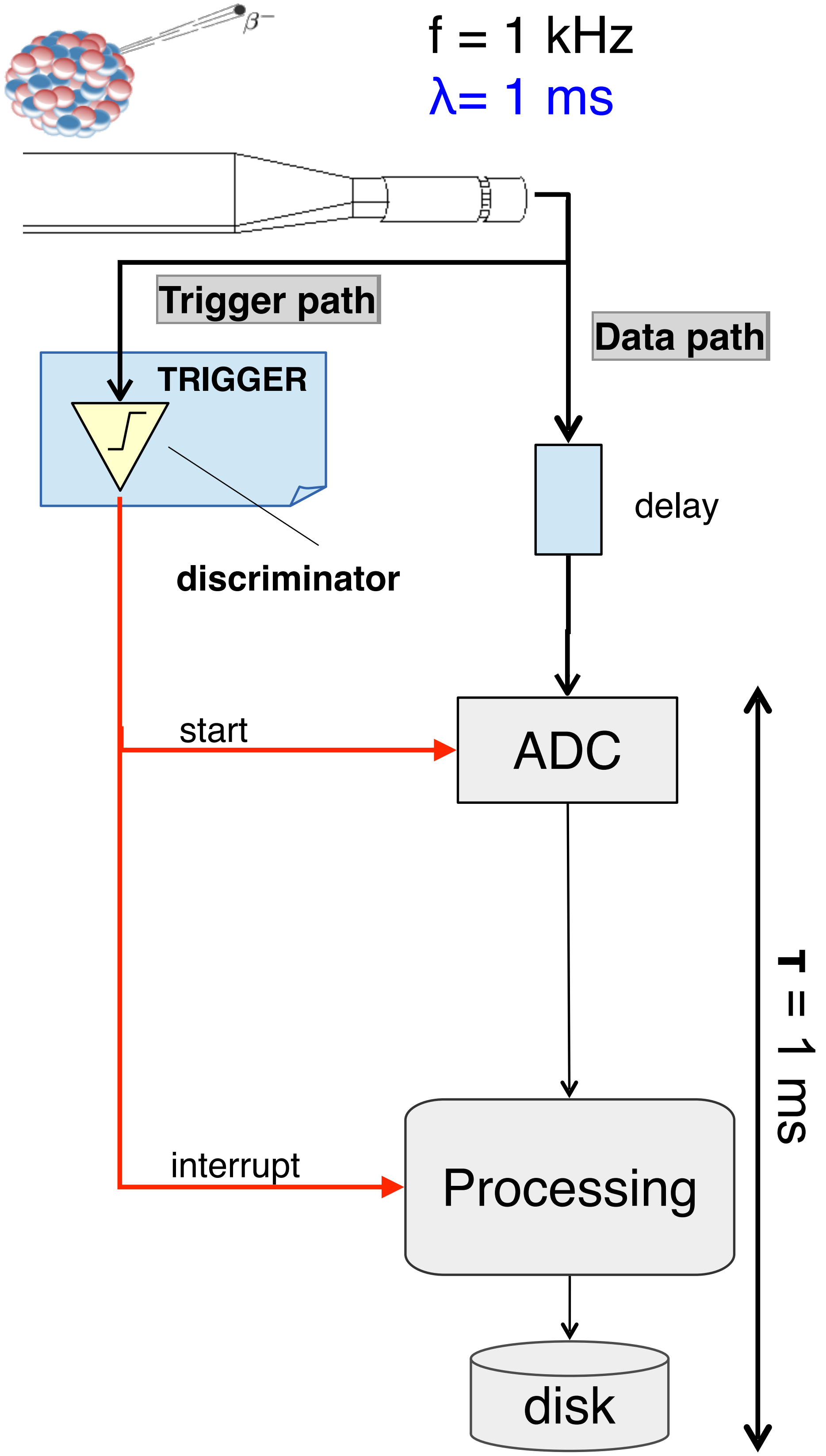
BUSY LOGIC

➡ Need a feedback mechanism, to know if the data processing pipeline is free to process a new event: the **busy logic**



BUSY LOGIC

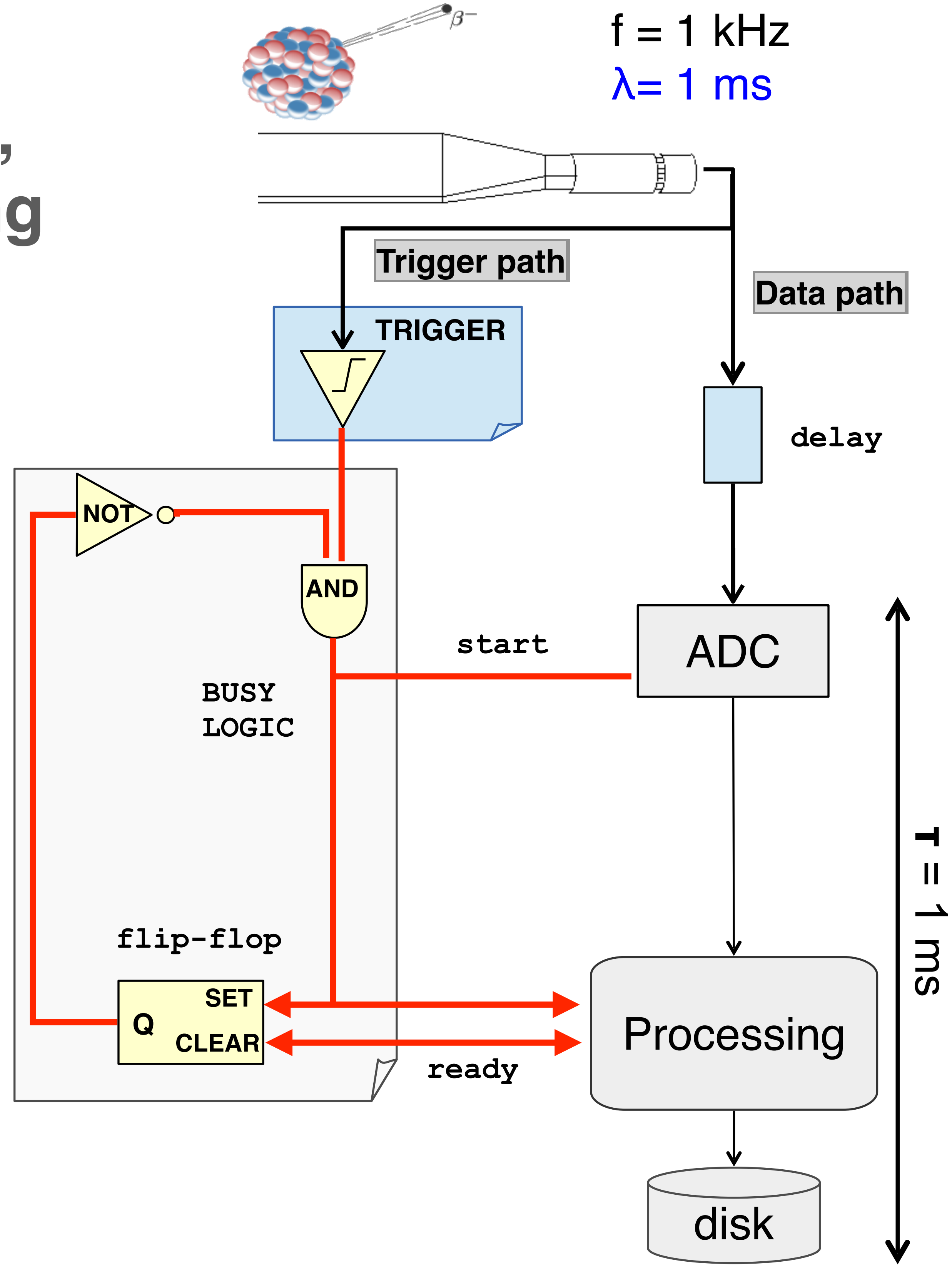
- **Need a feedback mechanism, to know if the data processing pipeline is free to process a new event: the busy logic**
- **A minimal busy logic can be implemented with**
  - an **AND** gate
  - a **NOT** gate
  - a **flip-flop**



BUSY LOGIC

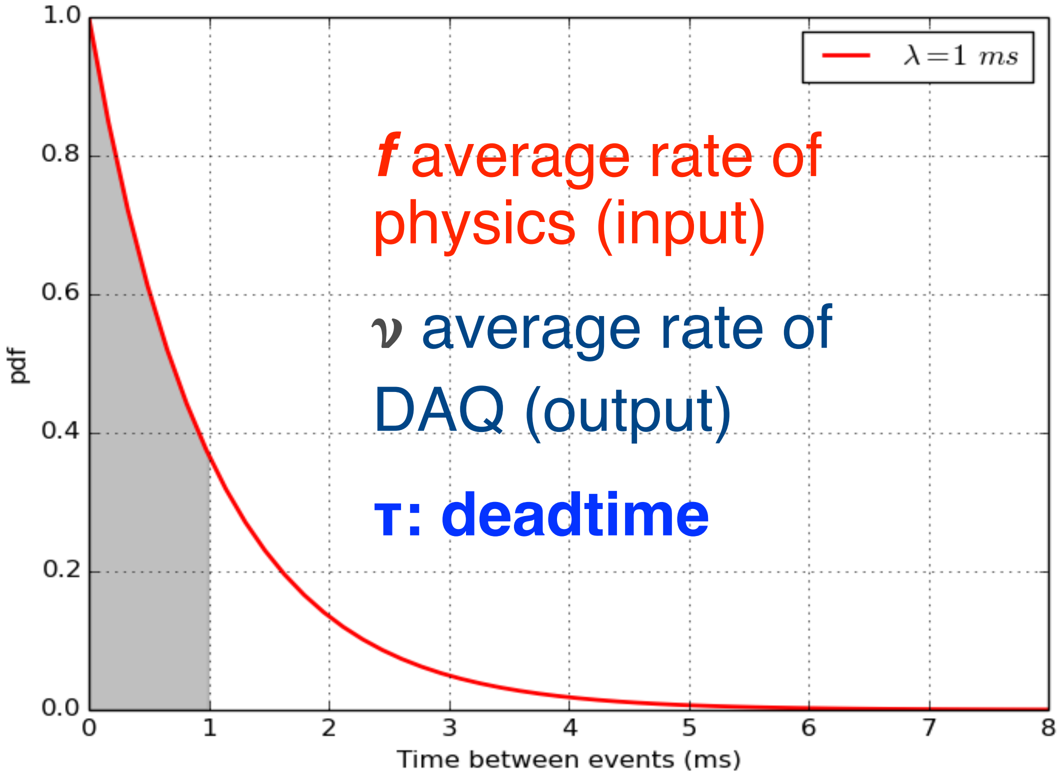
- Need a feedback mechanism, to know if the data processing pipeline is free to process a new event: the **busy logic**
- A minimal busy logic can be implemented with
  - an **AND** gate
  - a **NOT** gate
  - a **flip-flop**

Any new trigger is inhibited by the AND gate (busy)



DEADTIME AND EFFICIENCY

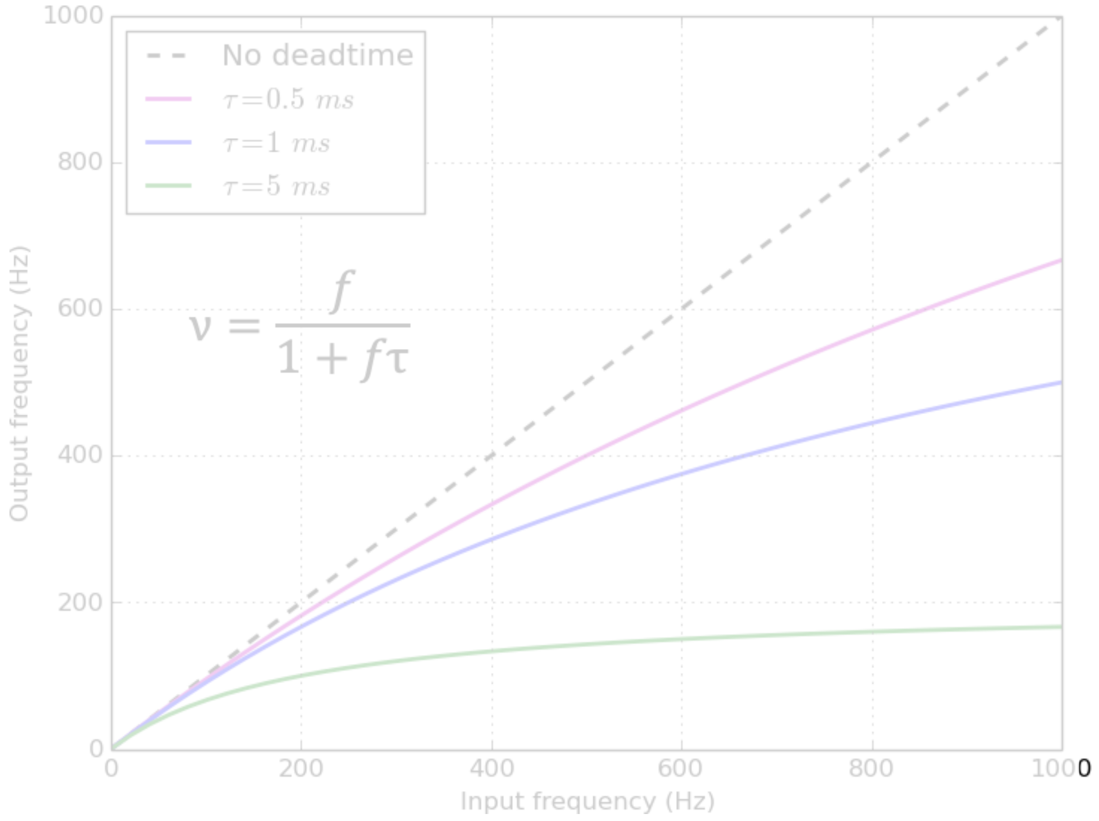
- **The busy mechanism protects our electronics from unwanted triggers**
  - During the busy time,
    no signals are accepted,
    cause of **inefficiency**
  - this is a source of **dead-time**
- **Due to stochastic fluctuations**
  - DAQ rate always < physics rate
  - Efficiency always < 100%
- **To cope with the input signal fluctuations, we have to over-design our DAQ system**
  - can we mitigate this effect?



f average rate of physics (input)

ν average rate of DAQ (output)

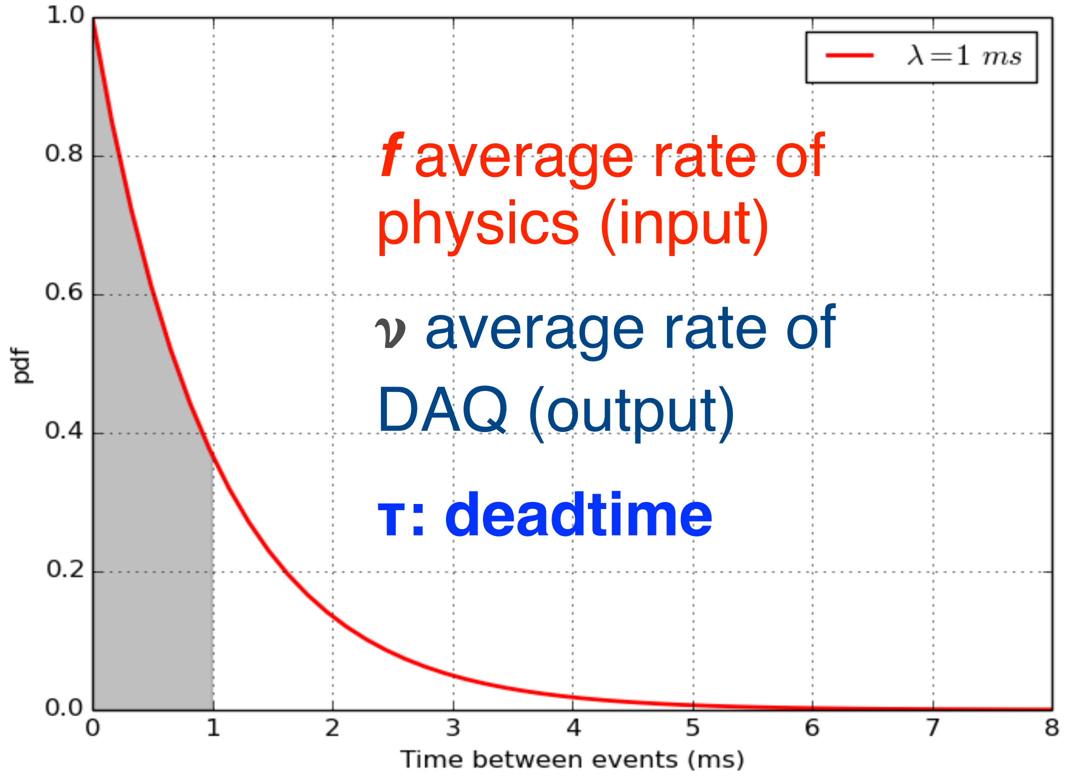
τ: deadtime



$$\nu = \frac{f}{1 + f\tau}$$

DEADTIME AND EFFICIENCY

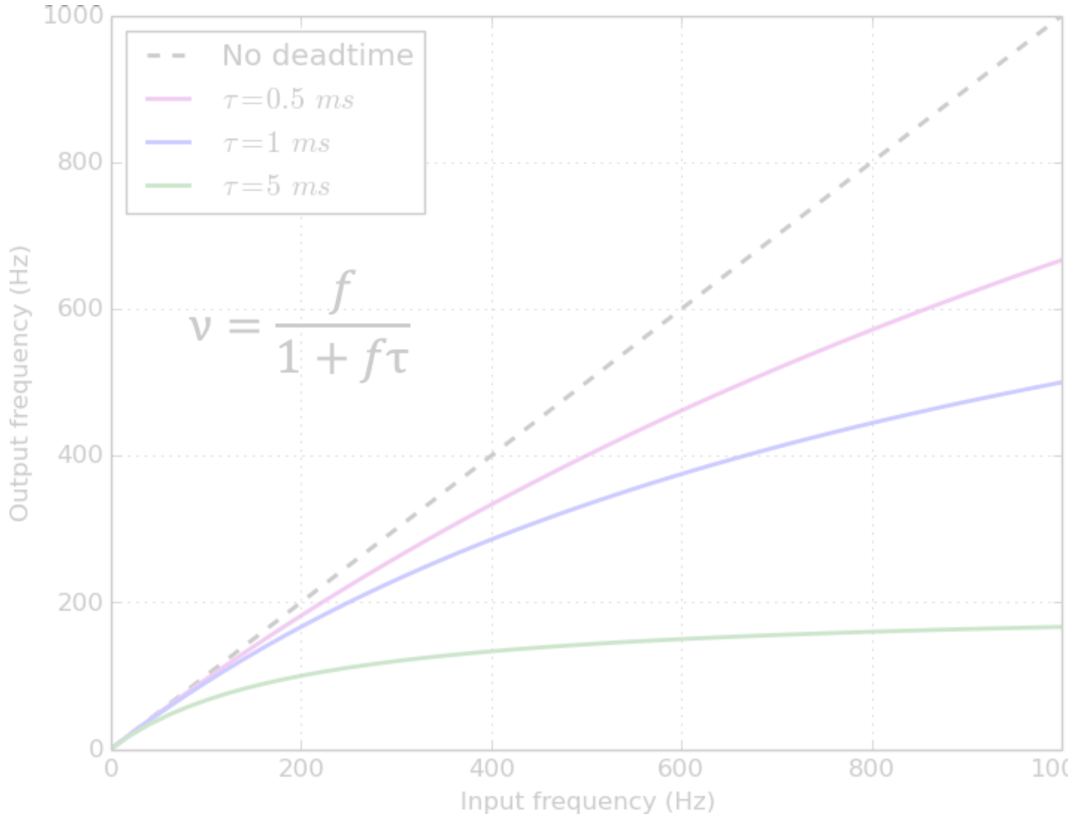
- ➡ **The busy mechanism protects our electronics from unwanted triggers**
  - ➡ During the busy time,
    no signals are accepted,
    cause of **inefficiency**
  - ➡ this is a source of **dead-time**
- ➡ **Due to stochastic fluctuations**
  - ➡ DAQ rate always < physics rate
  - ➡ Efficiency always < 100%
- ➡ **To cope with the input signal fluctuations, we have to over-design our DAQ system**
  - ➡ can we mitigate this effect?



f average rate of physics (input)

ν average rate of DAQ (output)

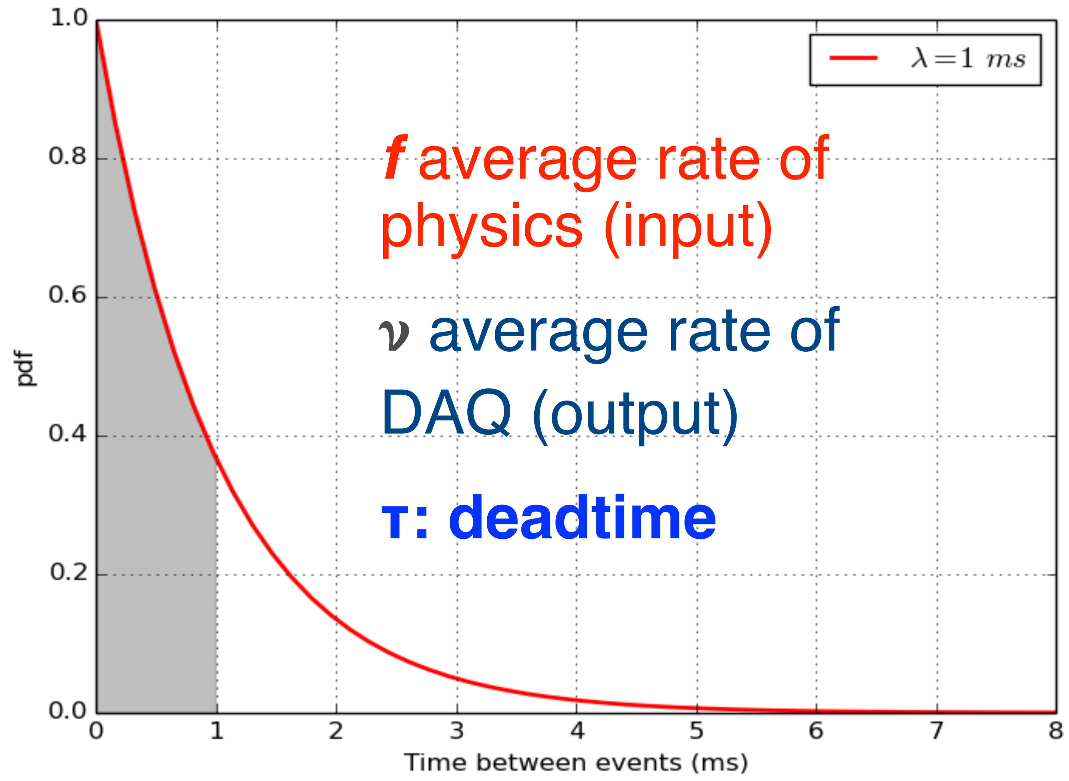
τ: deadtime



$$\nu = \frac{f}{1+f\tau}$$

DEADTIME AND EFFICIENCY

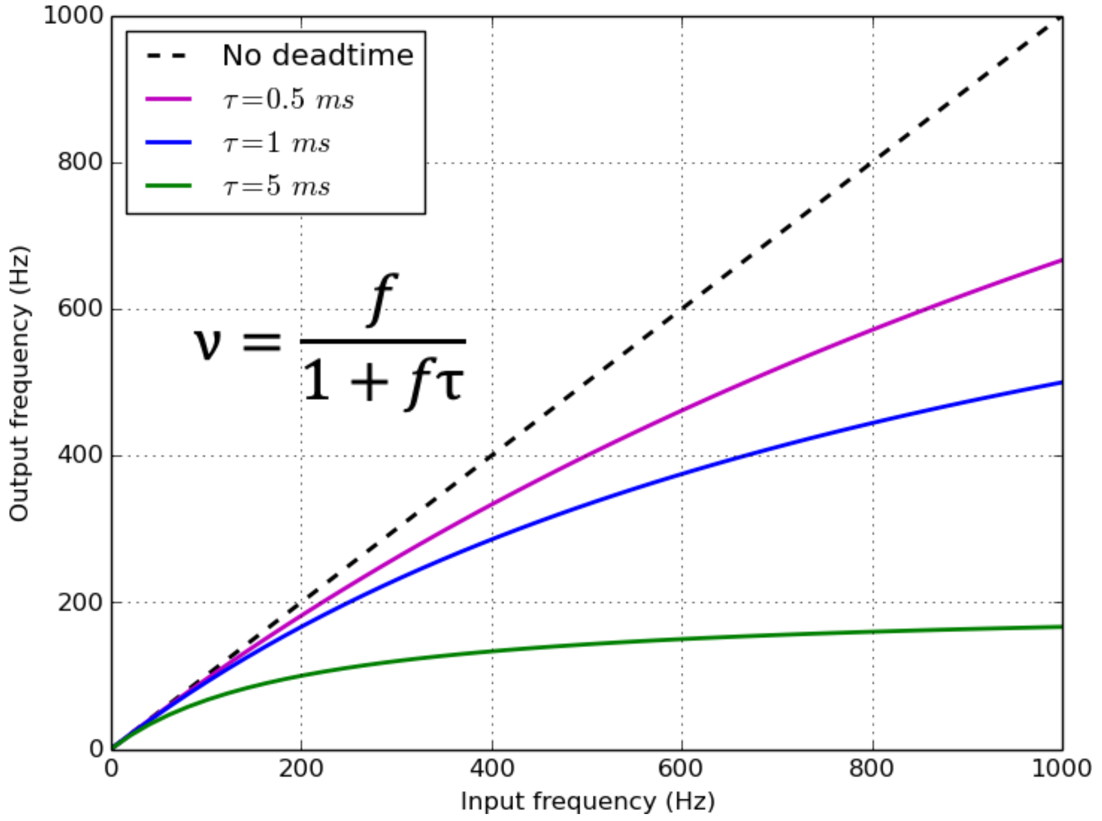
- **The busy mechanism protects our electronics from unwanted triggers**
  - During the busy time, no signals are accepted, cause of **inefficiency**
  - this is a source of **dead-time**
- **Due to stochastic fluctuations**
  - DAQ rate always < physics rate
  - Efficiency always < 100%
- **To cope with the input signal fluctuations, we have to over-design our DAQ system**
  - **can we mitigate this effect?**



f average rate of physics (input)

ν average rate of DAQ (output)

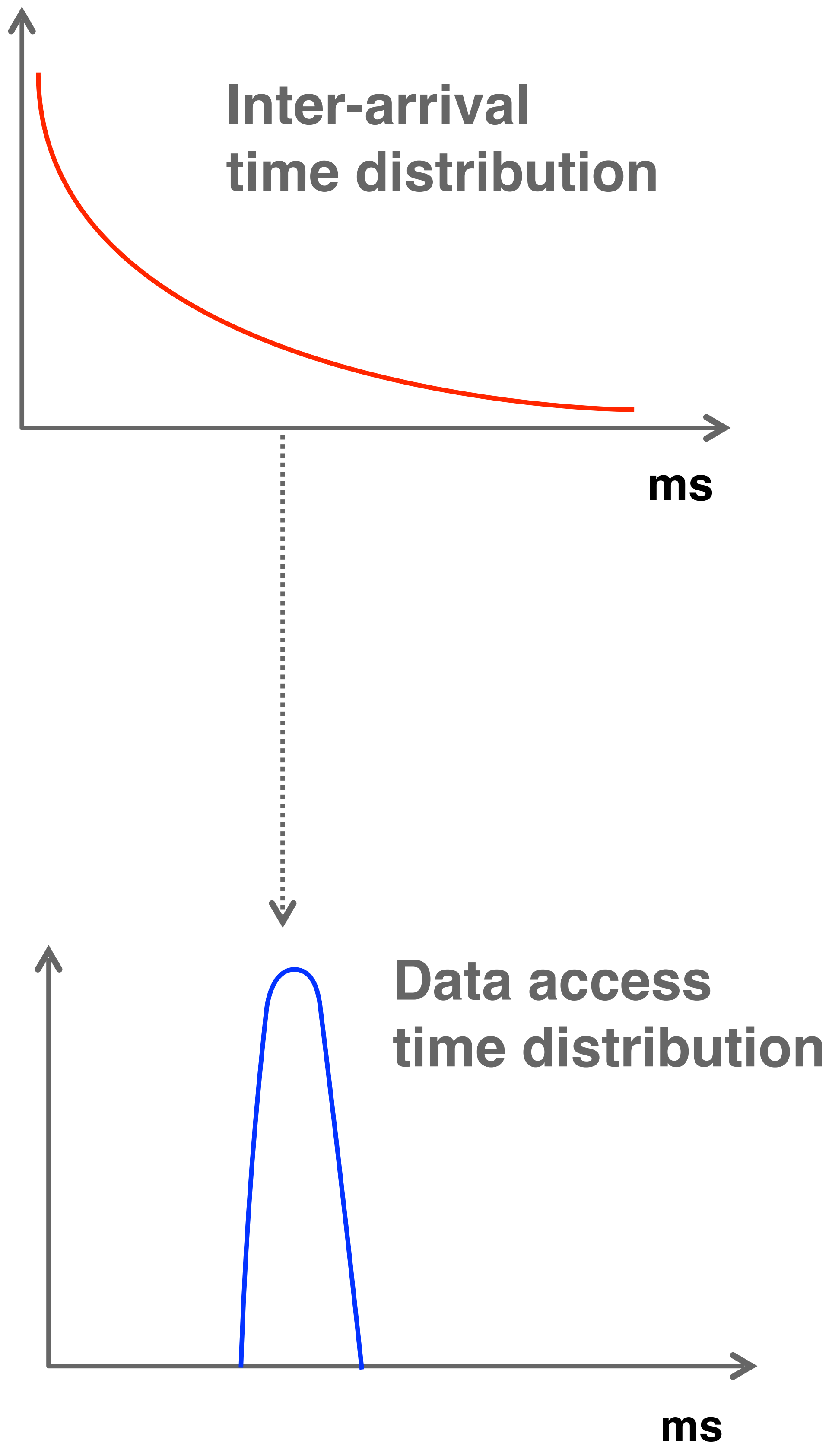
τ: deadtime



$$\nu = \frac{f}{1 + f\tau}$$

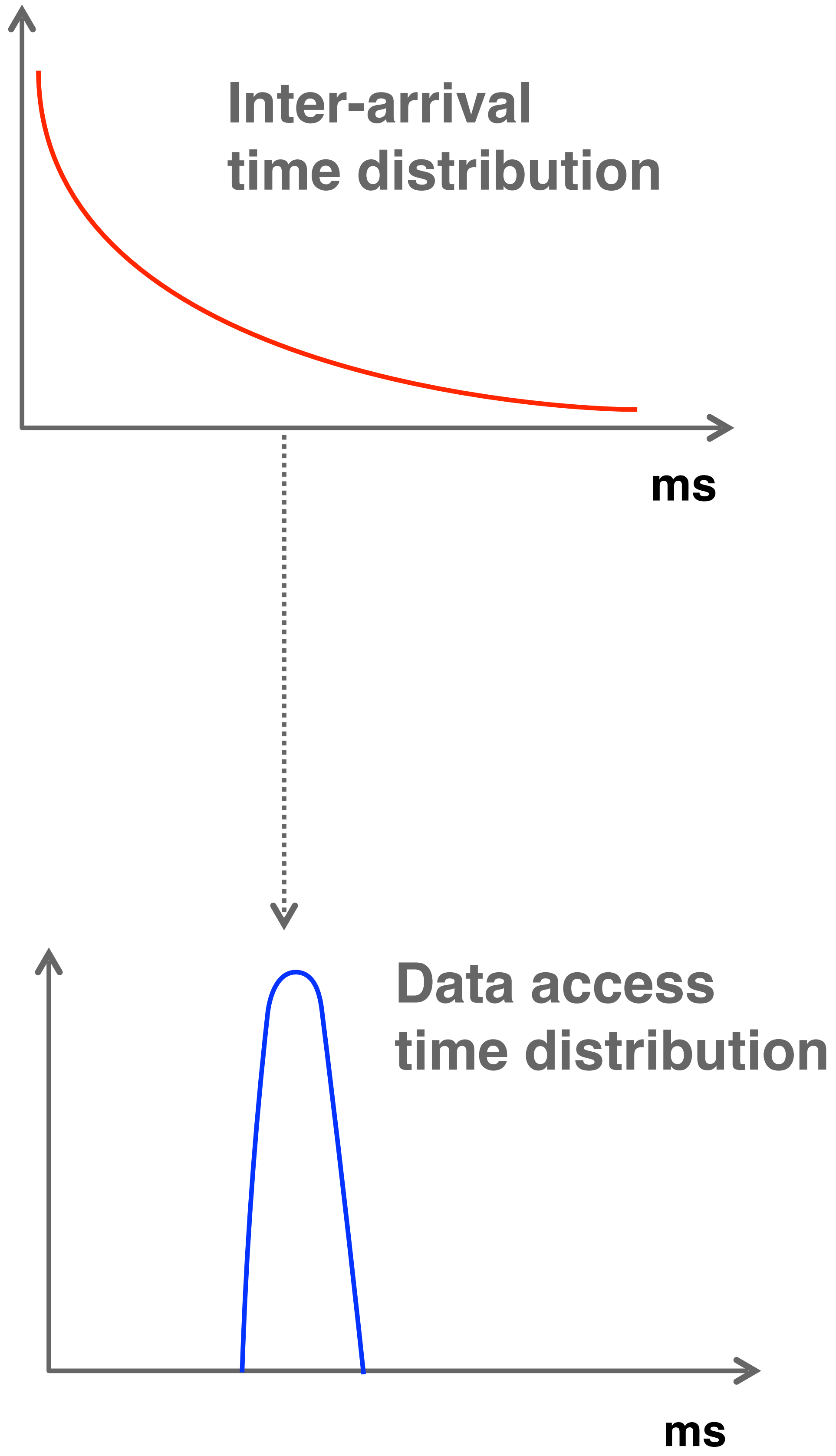
DE-RANDOMIZATION

- ➡ **What if we were able to make the system more deterministic and 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**
- ➡ **How can be achieved?**
  - ➡ by **buffering** the data (having a holding queue where we can slot it up to be processed)
  - ➡ Maintaining τ ~ λ (traffic intensity), high efficiency can be obtained even with moderate depth of FIFOs



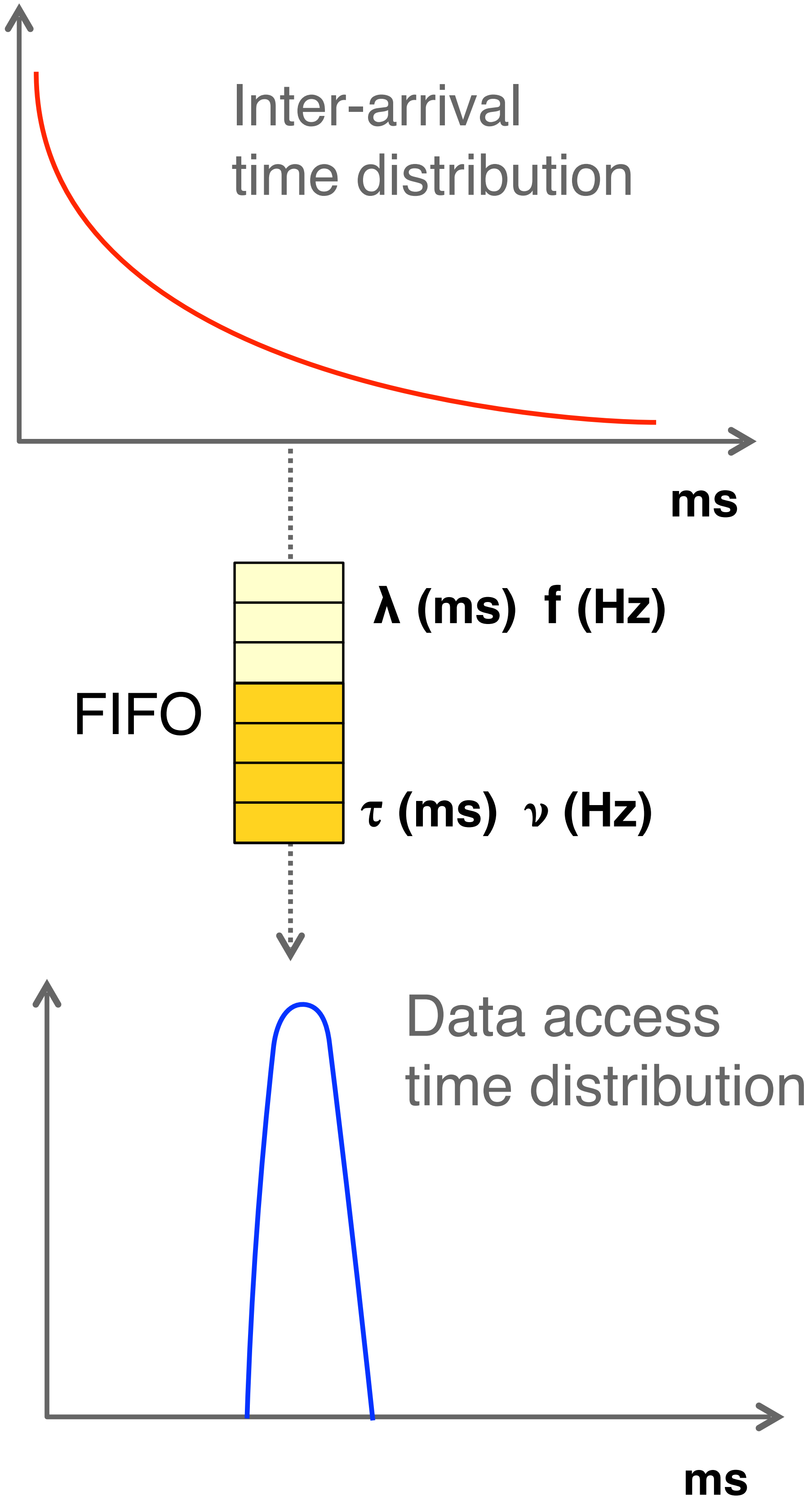
DE-RANDOMIZATION

- **What if we were able to make the system more deterministic and 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**
- **How can be achieved?**
  - by **buffering** the data (having a holding queue where we can slot it up to be processed)
  - Maintaining τ ~ λ (traffic intensity), high efficiency can be obtained even with moderate depth of FIFOs



DE-RANDOMIZATION

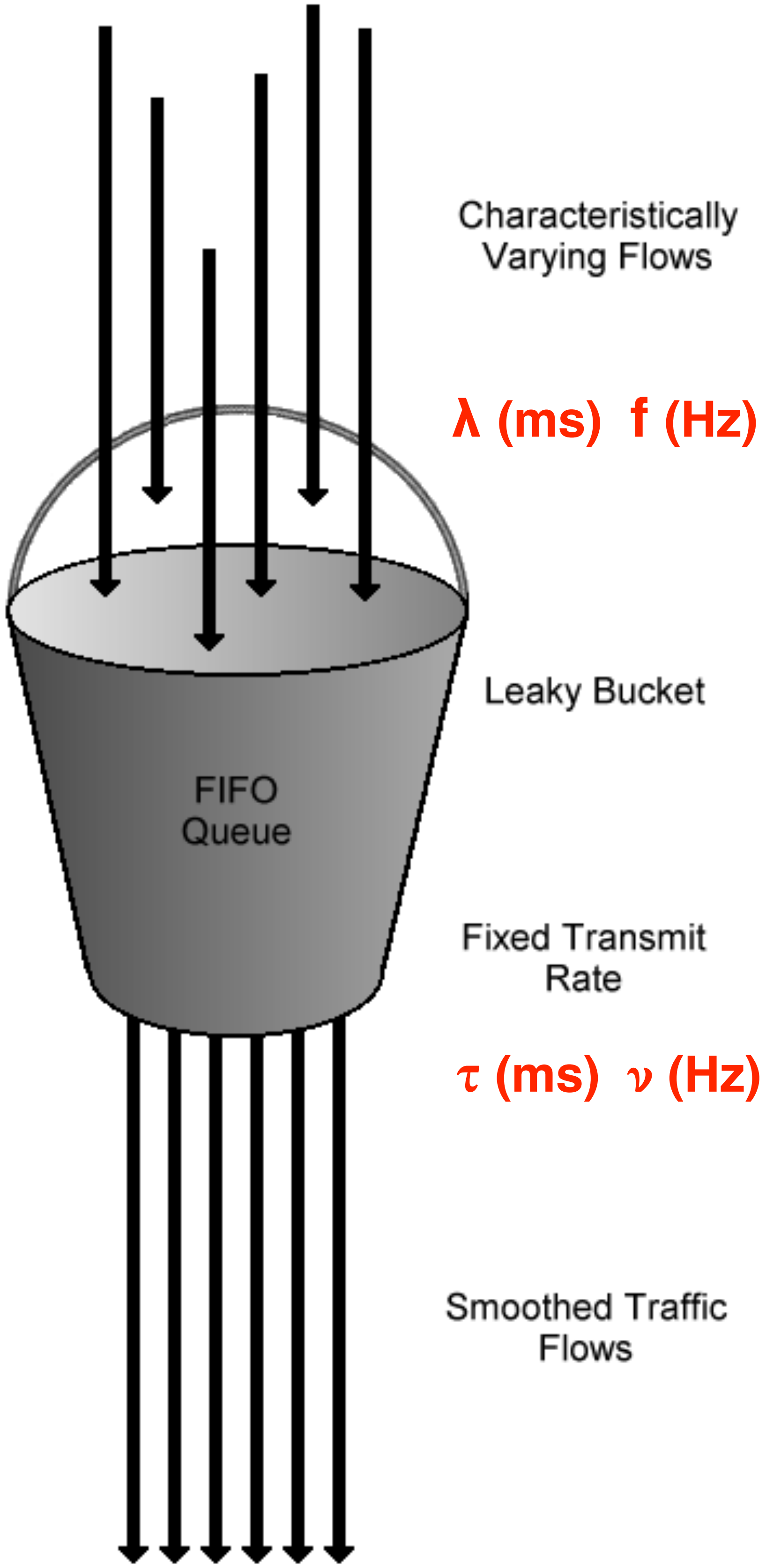
- **What if we were able to make the system more deterministic and less dependent on the arrival time of our signals?**
  - Then we could ensure that events don’t arrive when the system is busy
  - This is called **de-randomization**
- **How can be achieved?**
  - by **buffering** the data
  - having a holding **queue** where we can slot it up to be processed
  - Maintaining $\tau \sim \lambda$ **(traffic intensity)**, high efficiency can be obtained even with moderate depth of FIFOs



DE-RANDOMIZATION: THE LEAKY BUCKET

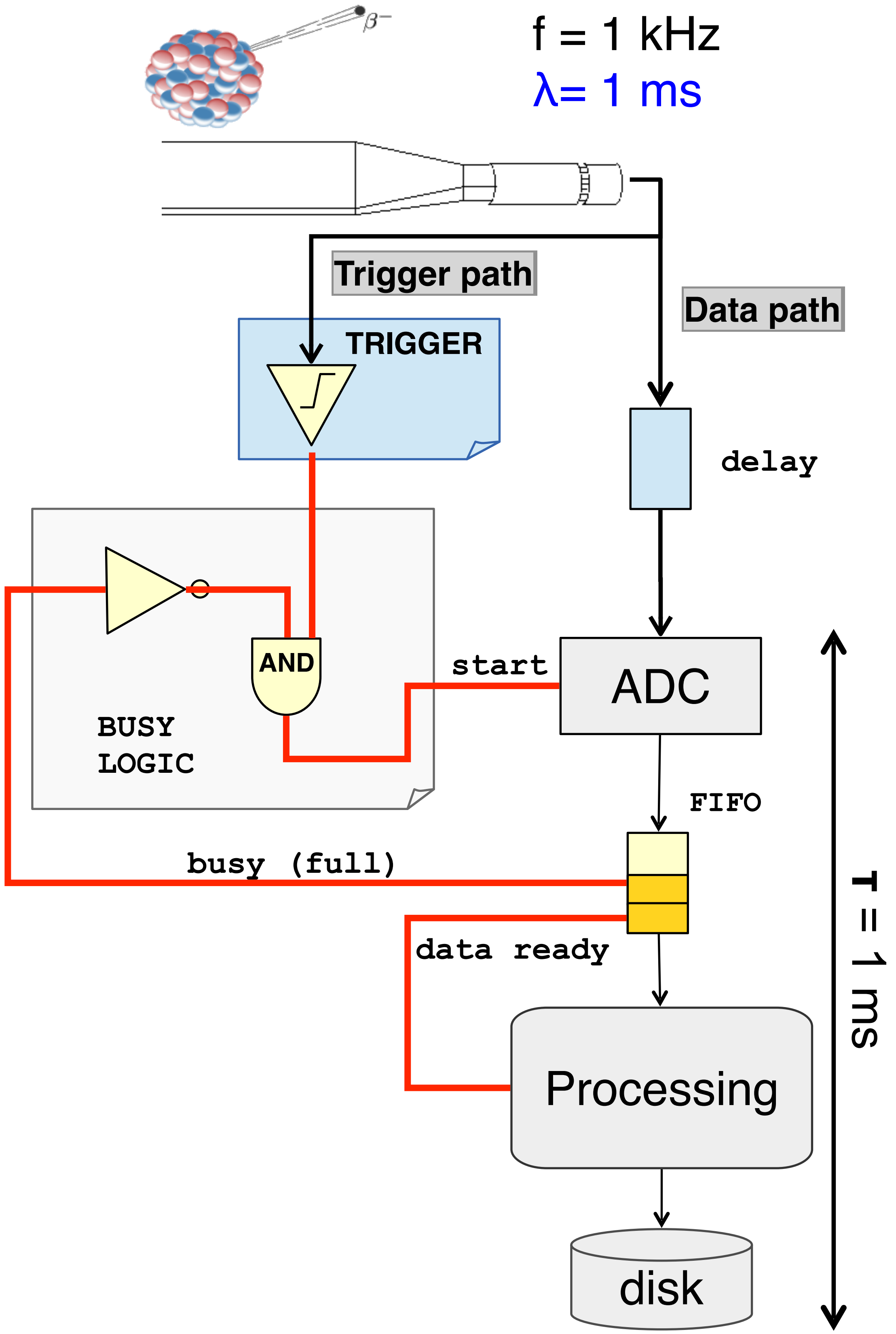
- **What if we were able to make the system more deterministic and less dependent on the arrival time of our signals?**
  - Then we could ensure that events don't arrive when the system is busy
  - This is called **de-randomization**
- **How can be achieved?**
  - by **buffering** the data
  - having a holding **queue** where we can slot it up to be processed
  - Maintaining **$\tau \sim \lambda$ (traffic intensity~1)**, high efficiency can be obtained even with moderate depth of FIFOs

—> *search for "Queuing theory"*



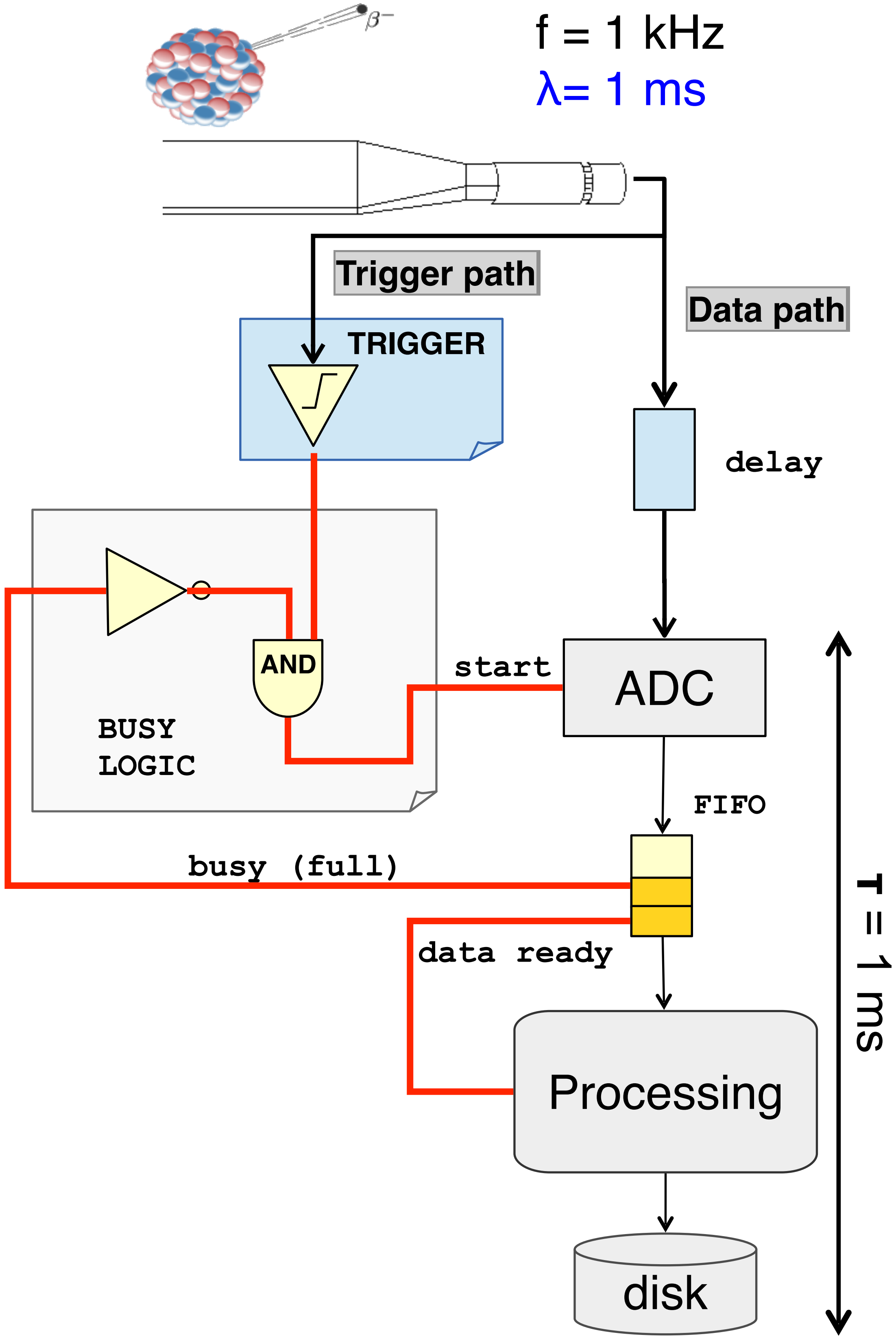
DE-RANDOMIZATION

- ➡ **Input fluctuations can be absorbed and smoothed by a queue**
  - ➡ A FIFO can provide a ~steady and **de-randomized** output rate
- ➡ **Busy is now defined by the buffer occupancy**
  - ➡ Processor pulls data from the buffer at fixed rate, separating the event receiving and data processing steps



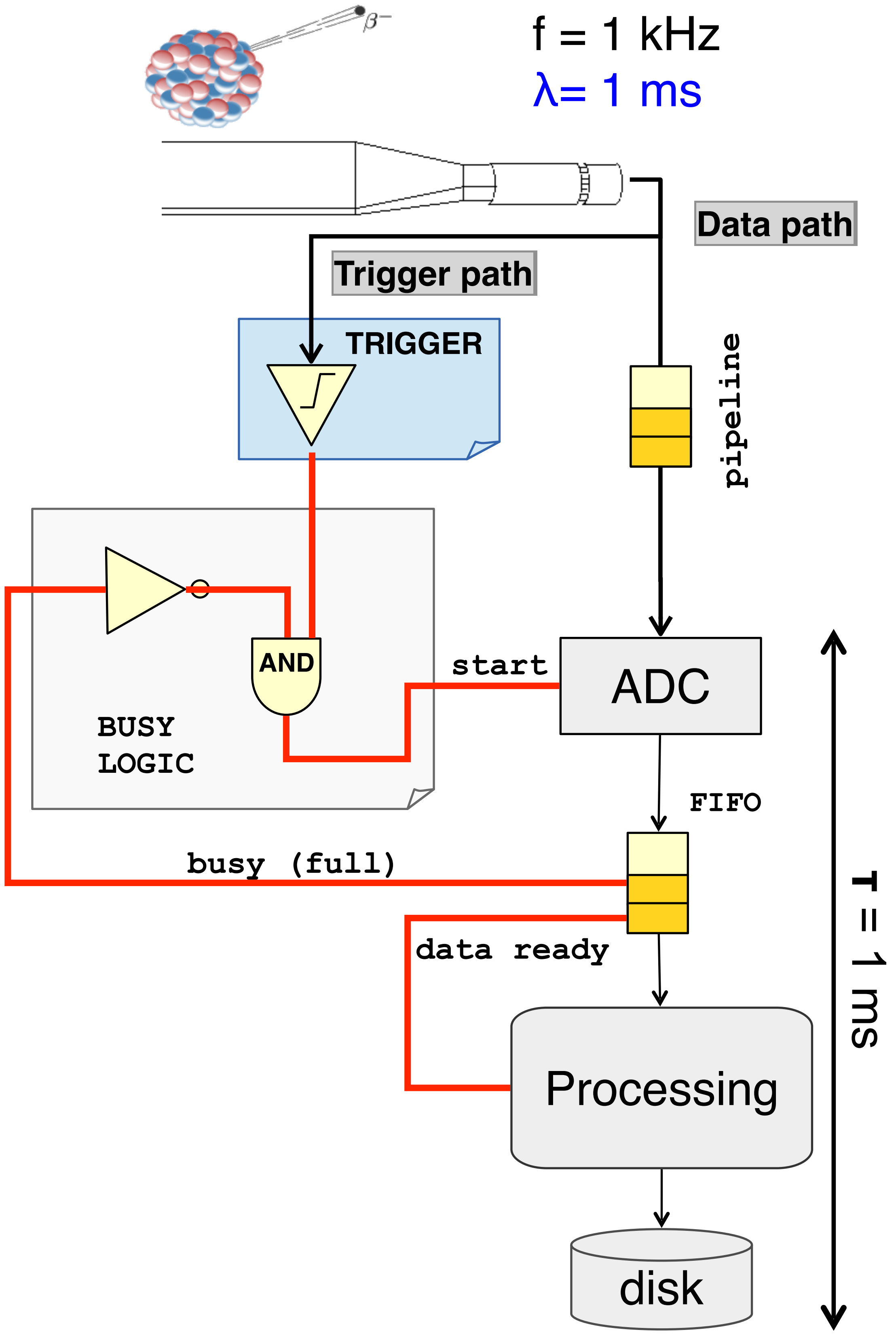
DE-RANDOMIZATION SUMMARY

- **The FIFO decouples the low latency front-end from the data processing**
    - Minimize the amount of “unnecessary” fast components
- **~100% efficiency with minimal deadtime achievable if**
    - ADC can operate at rate >> f
    - Data processing and storing operate at a rate ~ f
- **Could the delay be replaced with a “FIFO”?**
    - Analog pipelines, heavily used in LHC DAQs

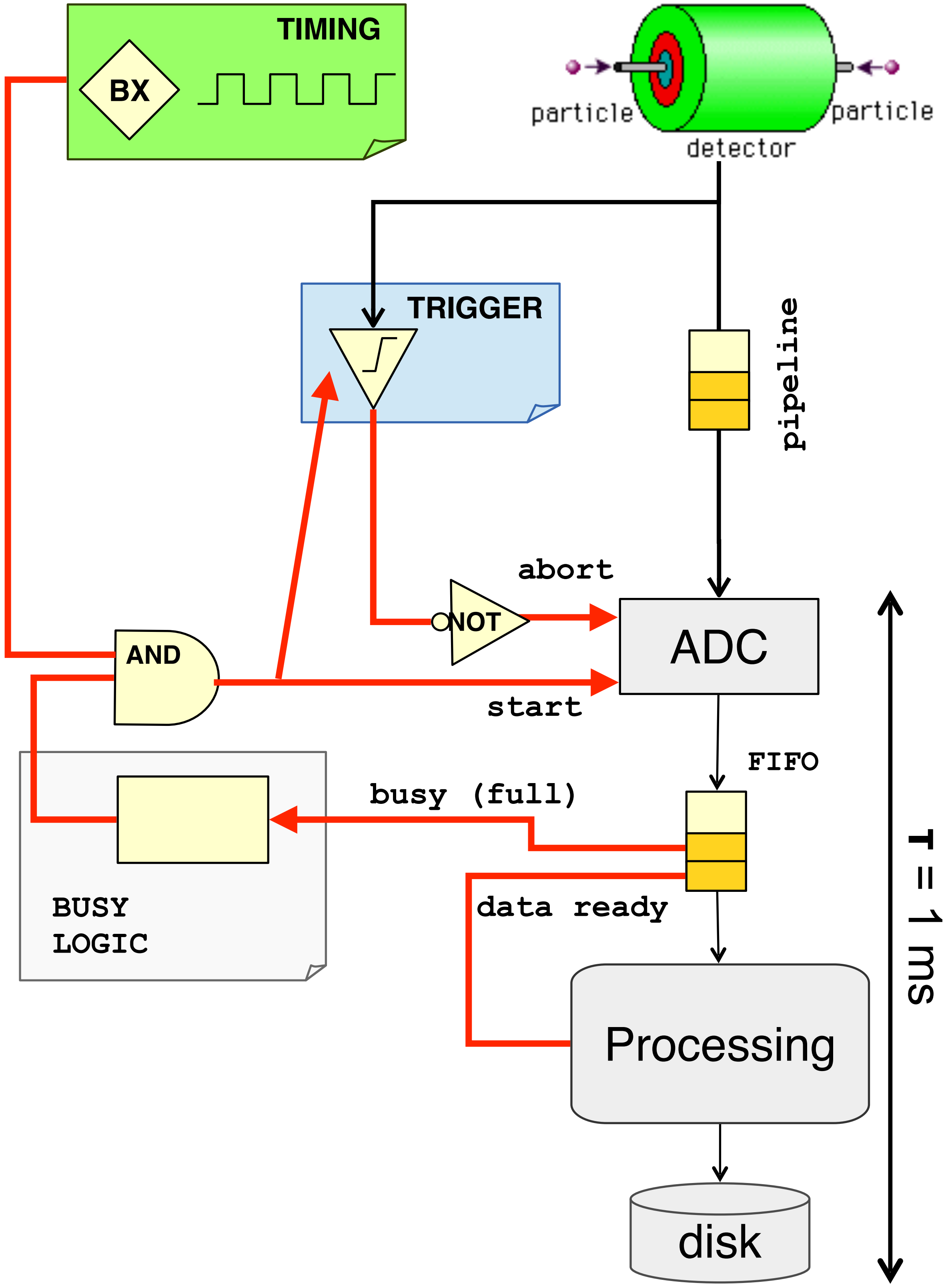


DE-RANDOMIZATION SUMMARY

- **The FIFO decouples the low latency front-end from the data processing**
  - Minimize the amount of “unnecessary” fast components
- **~100% efficiency with minimal deadtime achievable if**
  - ADC can operate at rate >> f
  - Data processing and storing operate at a rate ~ f
- **Could the delay be replaced with a “FIFO”?**
  - Analog pipelines, heavily used in LHC DAQs

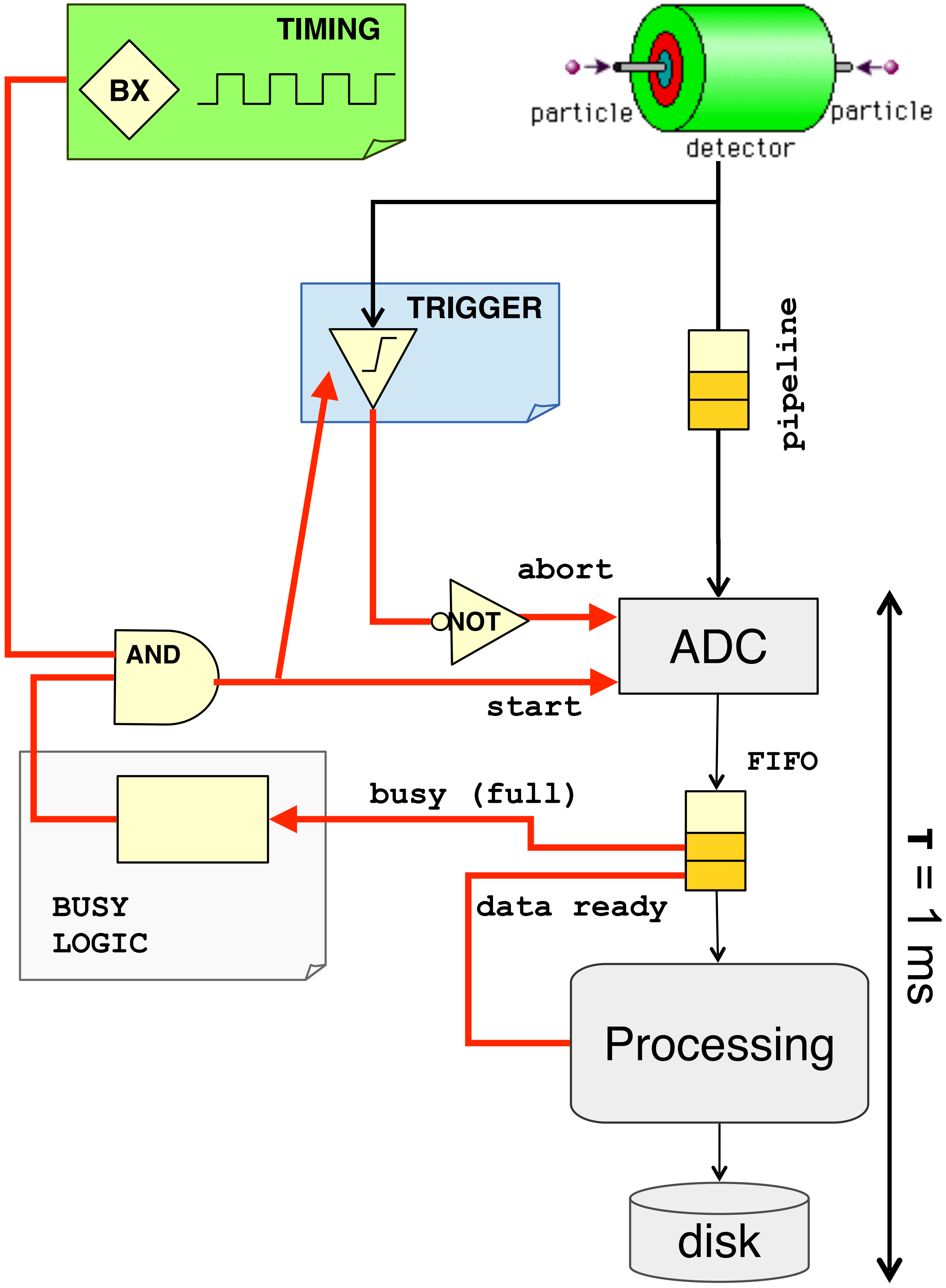


COLLIDER SET-UP



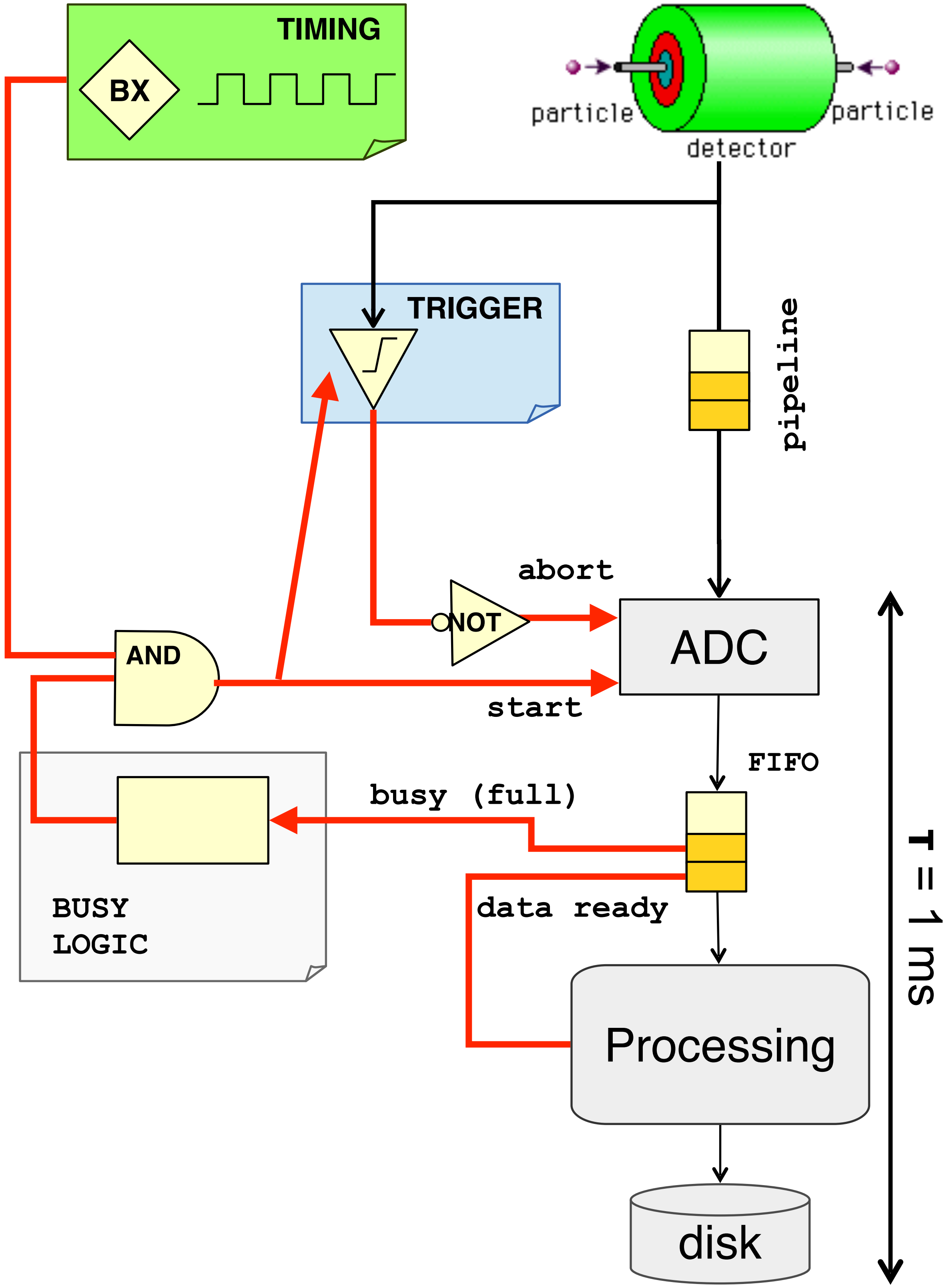
COLLIDER SET-UP

- ➡ **Do we need de-randomization buffers also in collider setups?**
  - ➡ Particle collisions are synchronous
  - ➡ But the time distribution of triggers is random: good events are unpredictable



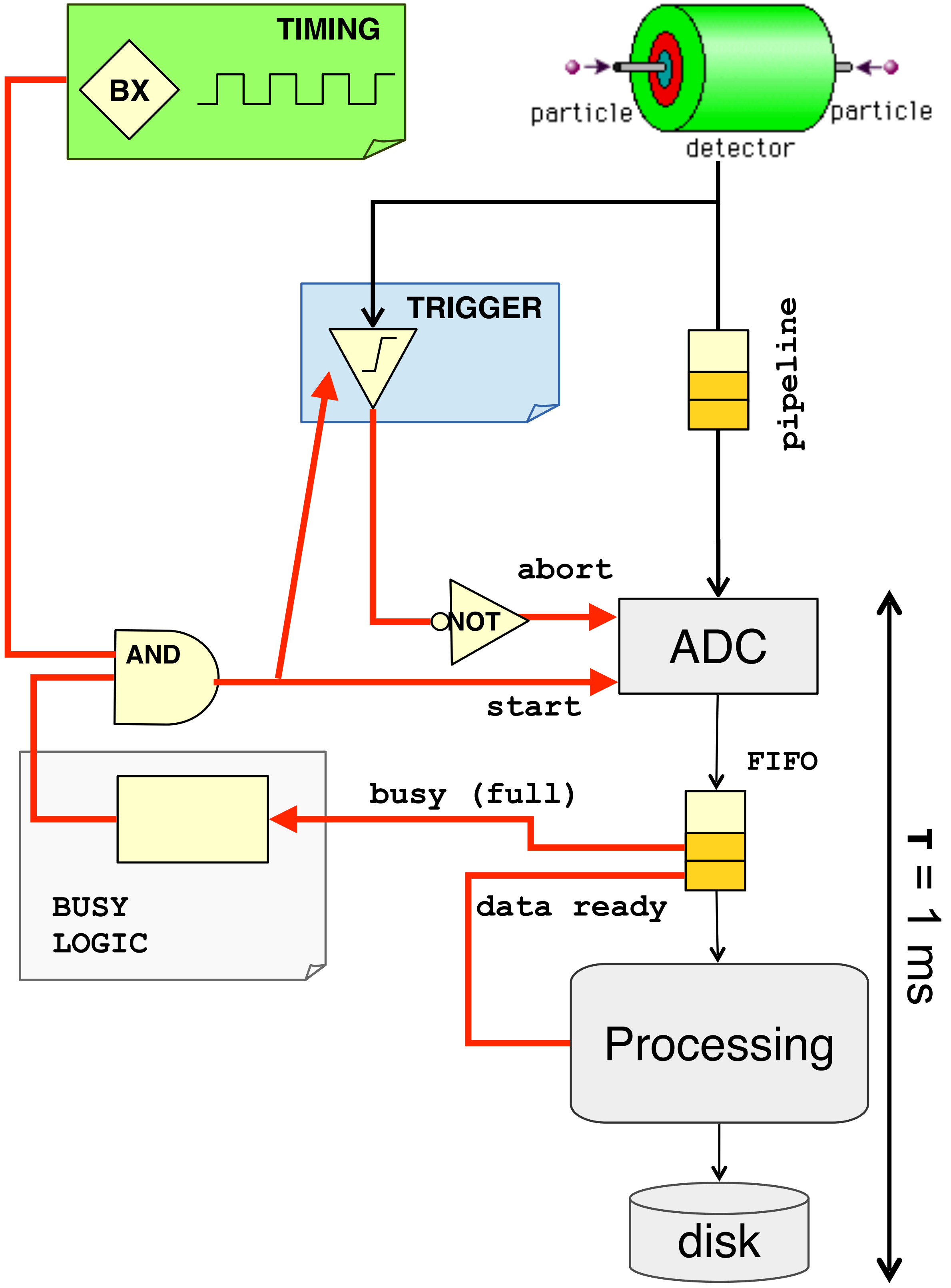
COLLIDER SET-UP

- ➡ **Do we need de-randomization buffers also in collider setups?**
  - ➡ Particle collisions are synchronous
  - ➡ But the time distribution of triggers is random: good events are unpredictable
- ➡ **De-randomization is still needed**



COLLIDER SET-UP

- ➡ **Do we need de-randomization buffers also in collider setups?**
  - ➡ Particle collisions are synchronous
  - ➡ But the time distribution of triggers is random: good events are unpredictable
- ➡ **De-randomization is still needed**
- ➡ **More complex busy logic to protect buffers and detectors**
  - ➡ Eg: accept n events every m bunch crossings
  - ➡ Eg: prevent some dangerous trigger patterns

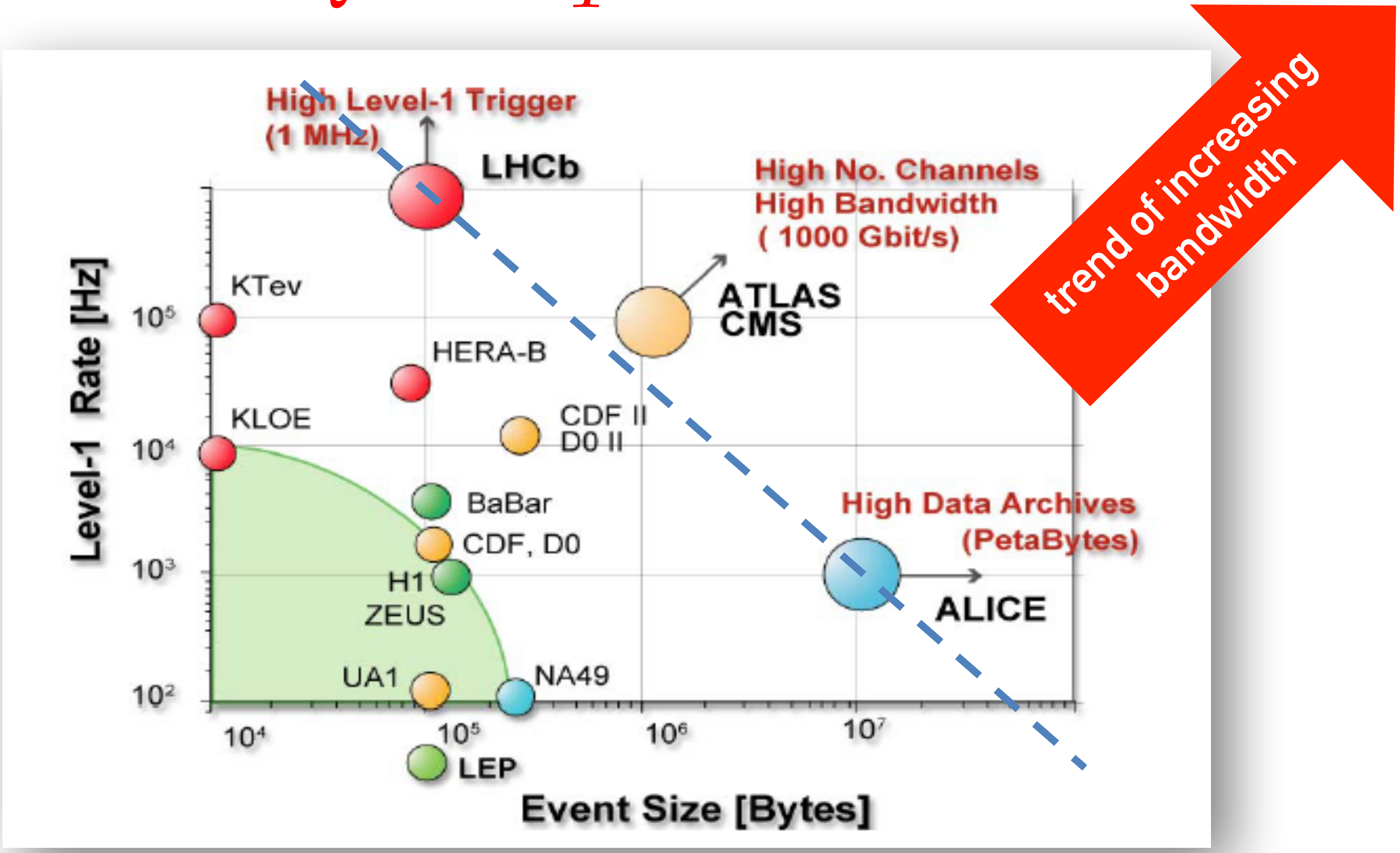


OUTLINE

- ➡ **Introduction**
  - ➡ What is Trigger and DAQ?
  - ➡ Overall TDAQ framework
- ➡ **Basic TDAQ concepts**
  - ➡ Digitization, Latency
  - ➡ Deadtime, Busy
  - ➡ De-randomization
- ➡ **Scaling up**
  - ➡ Readout and Event Building
  - ➡ Buses vs Network
- ➡ **Fight bottlenecks**

READOUT AND DAQ THROUGHPUTS

$$R_{DAQ} = R_T^{max} \times S_E$$

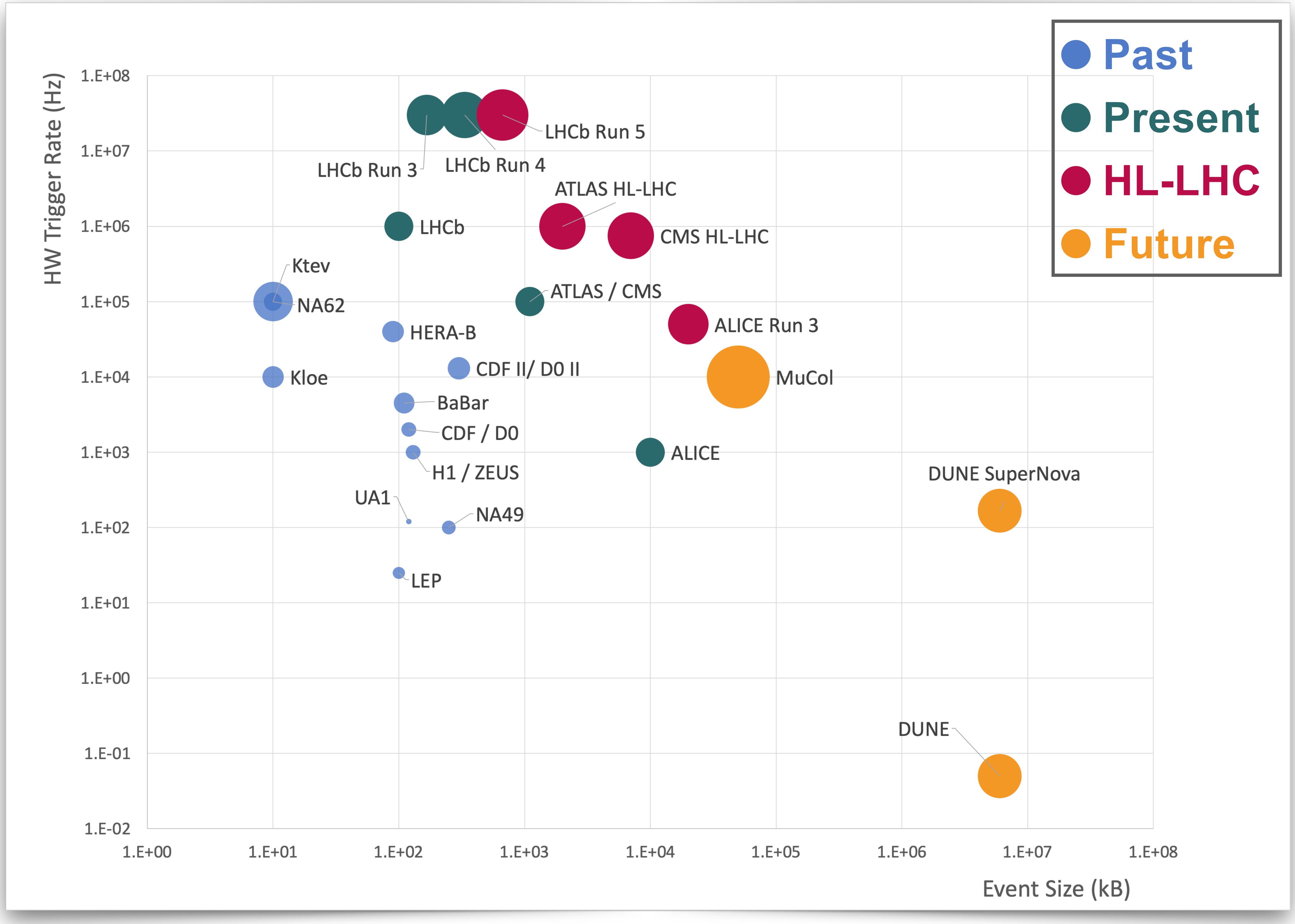


faster L1 electronics

more channels, more complex events

➡ **As the data volumes and rates increase, new architectures are needed**

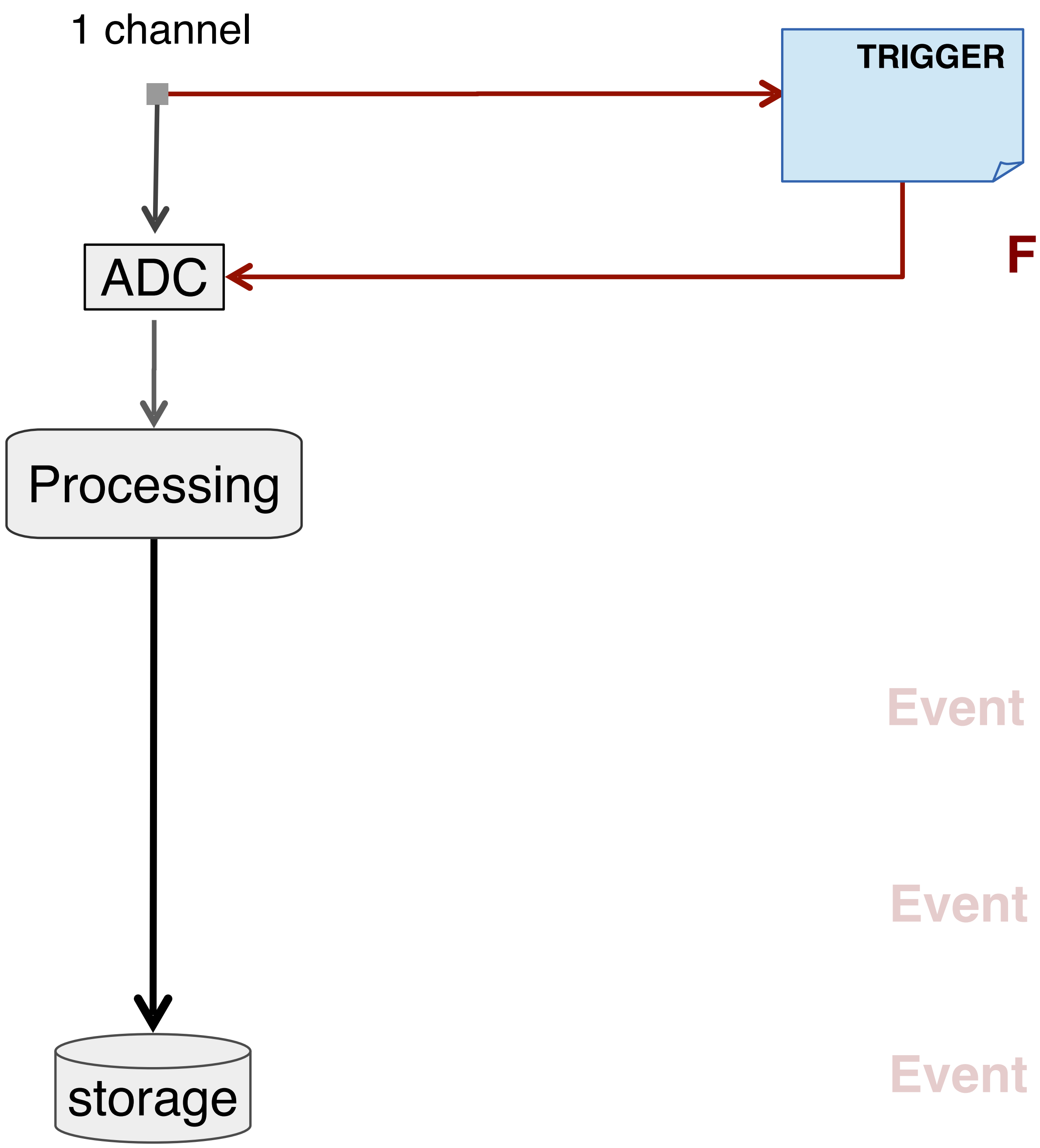
UPDATED FIGURE!



Courtesy of A.Cerri

ADDING MORE CHANNELS

- **Adding more channels requires a hierarchical structure committed to the data handling and conveyance**



Front-End | data digitization
data buffering

Readout | data extraction
data formatting
data buffering

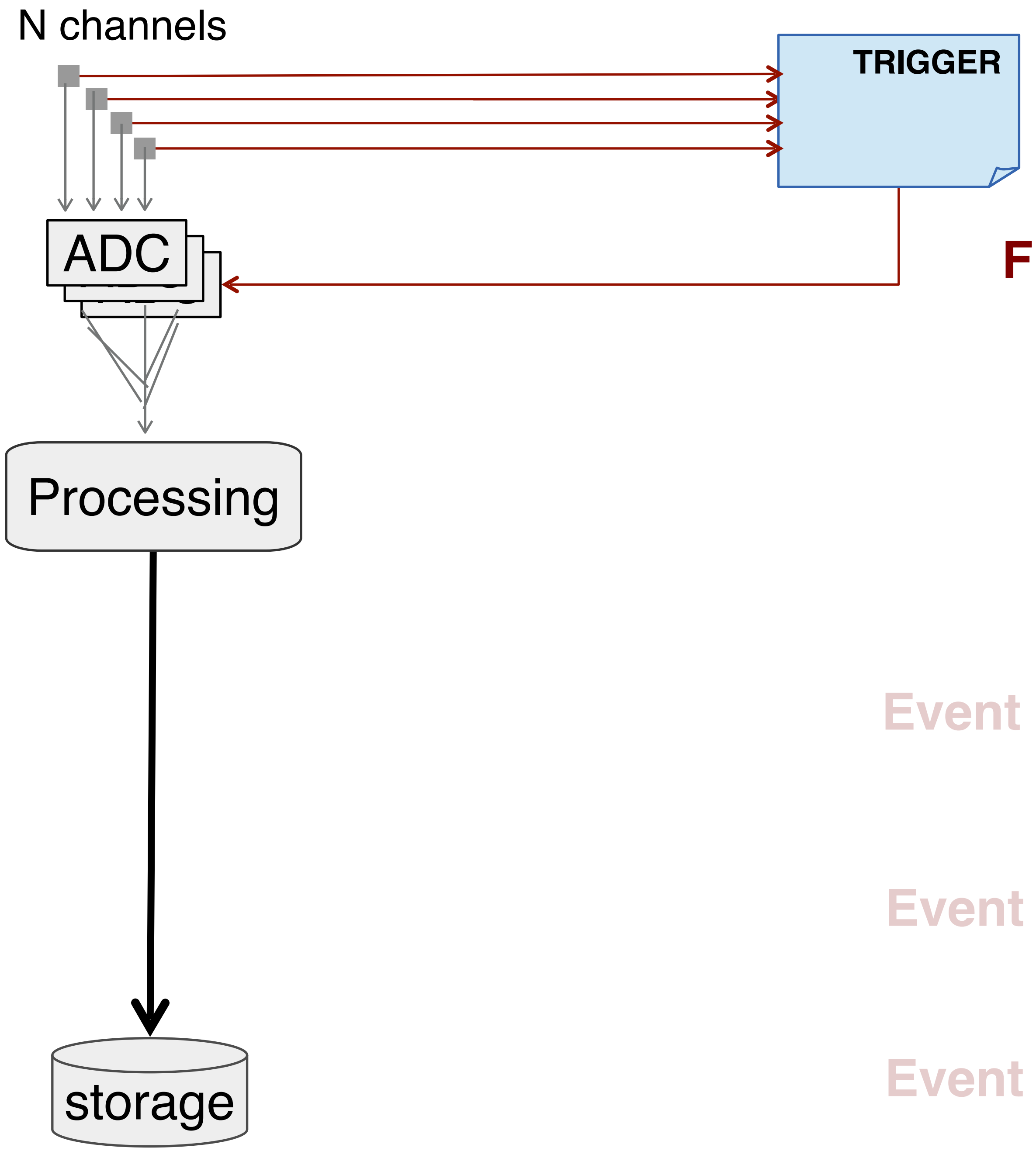
Event Building | event assembly
event buffering

Event Filtering | event rejection
event buffering

Event Logging | file storage
file buffering

ADDING MORE CHANNELS

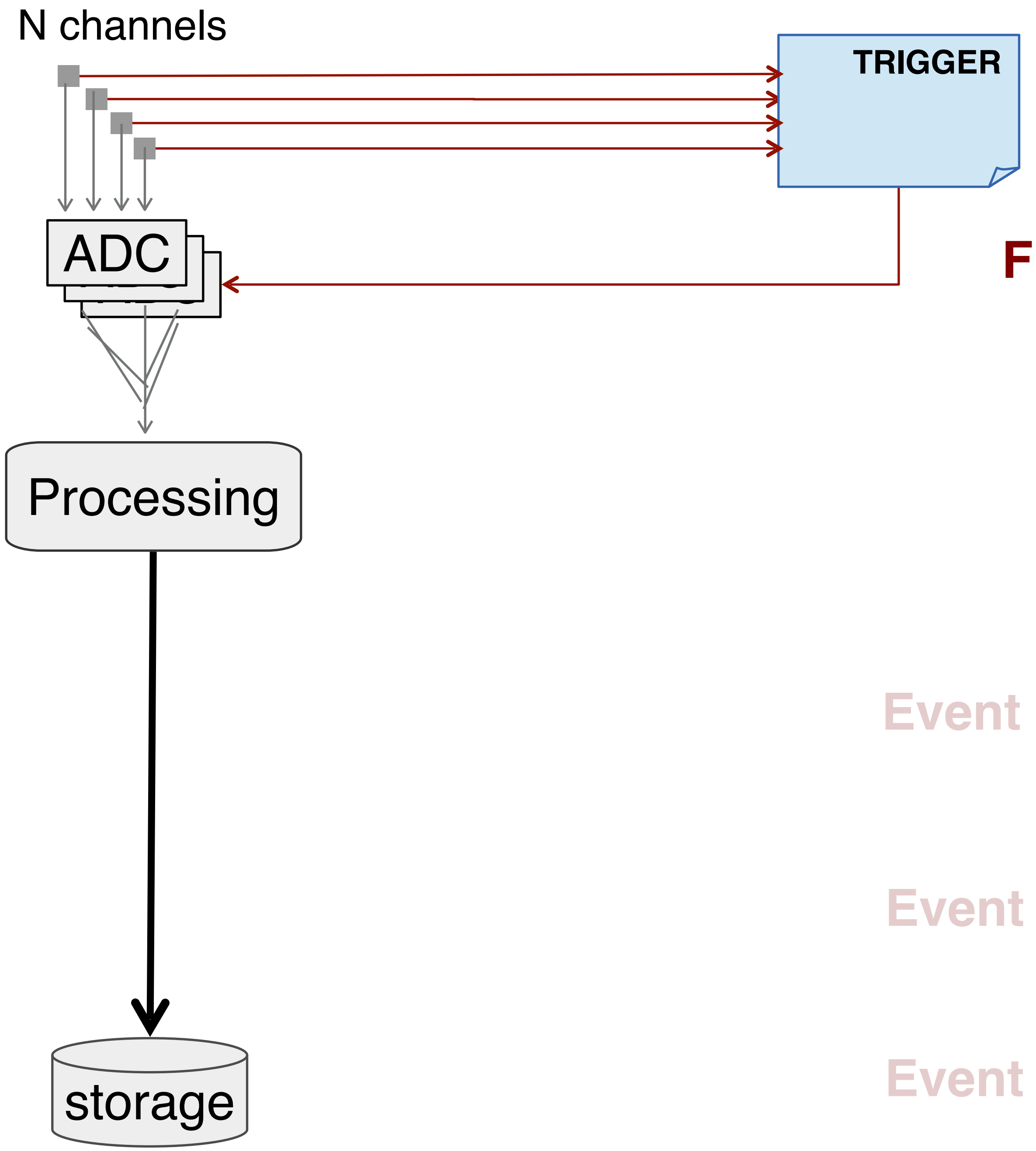
- **Adding more channels requires a hierarchical structure committed to the data handling and conveyance**



Front-End	data digitization data buffering
Readout	data extraction data formatting data buffering
Event Building	event assembly event buffering
Event Filtering	event rejection event buffering
Event Logging	file storage file buffering

ADDING MORE CHANNELS

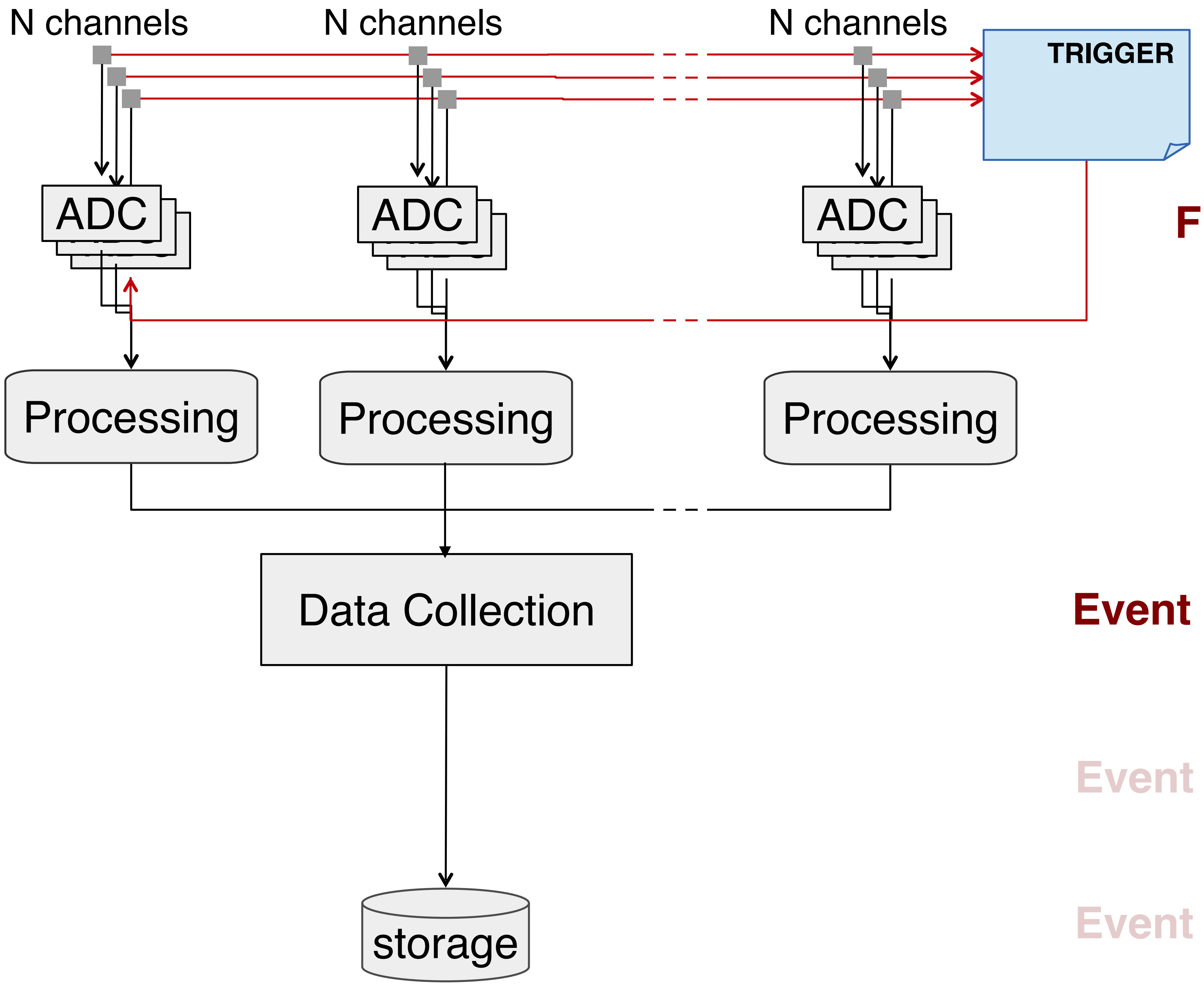
- **Adding more channels requires a hierarchical structure committed to the data handling and conveyance**



Trigger	clock synch trigger decision
Front-End	data digitization data buffering
Readout	data extraction data formatting data buffering
Event Building	event assembly event buffering
Event Filtering	event rejection event buffering
Event Logging	file storage file buffering

ADDING MORE CHANNELS

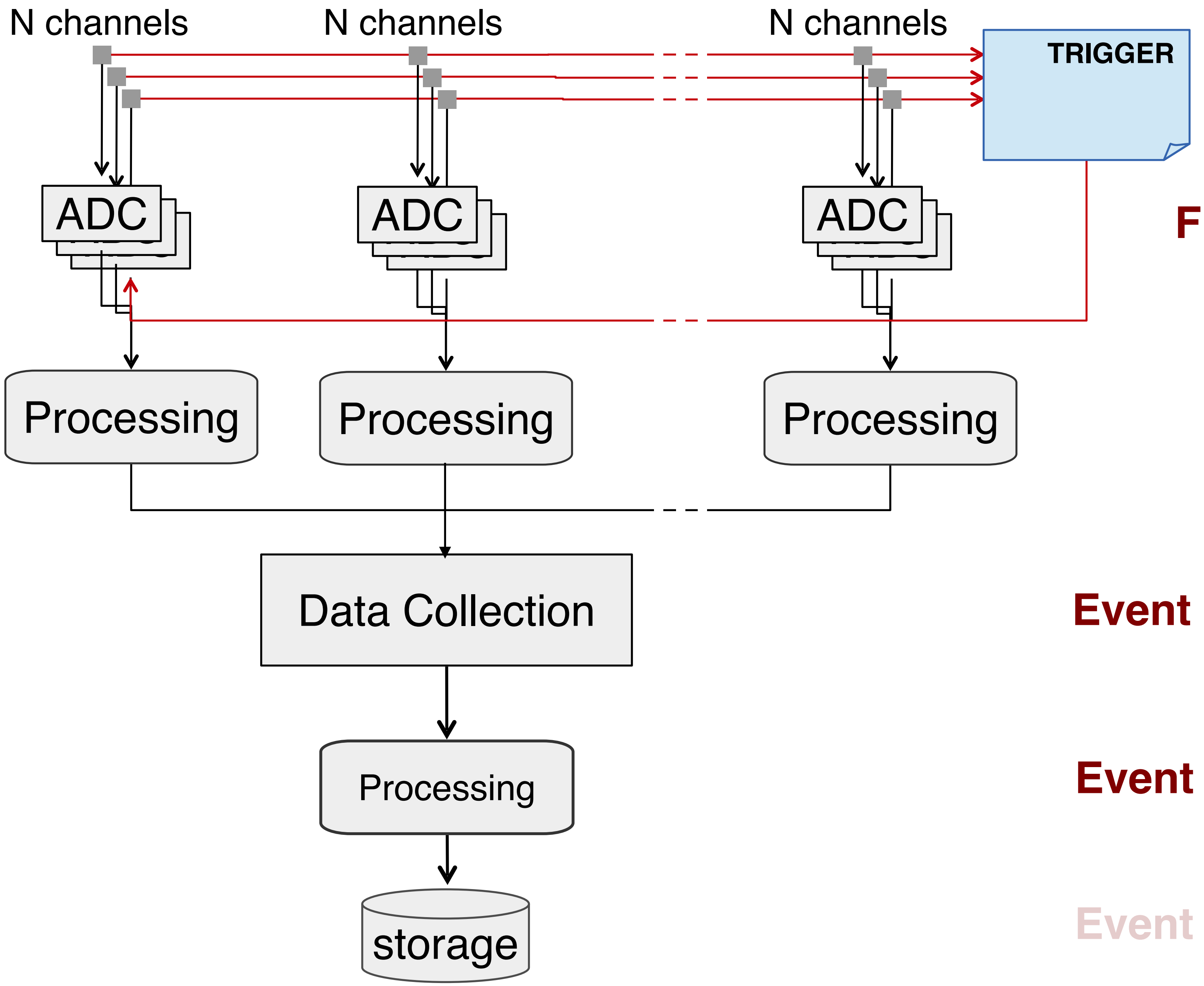
- **Adding more channels requires a hierarchical structure committed to the data handling and conveyance**



Trigger	clock synch trigger decision
Front-End	data digitization data buffering
Readout	data extraction data formatting data buffering
Event Building	event assembly event buffering
Event Filtering	event rejection event buffering
Event Logging	file storage file buffering

ADDING MORE CHANNELS

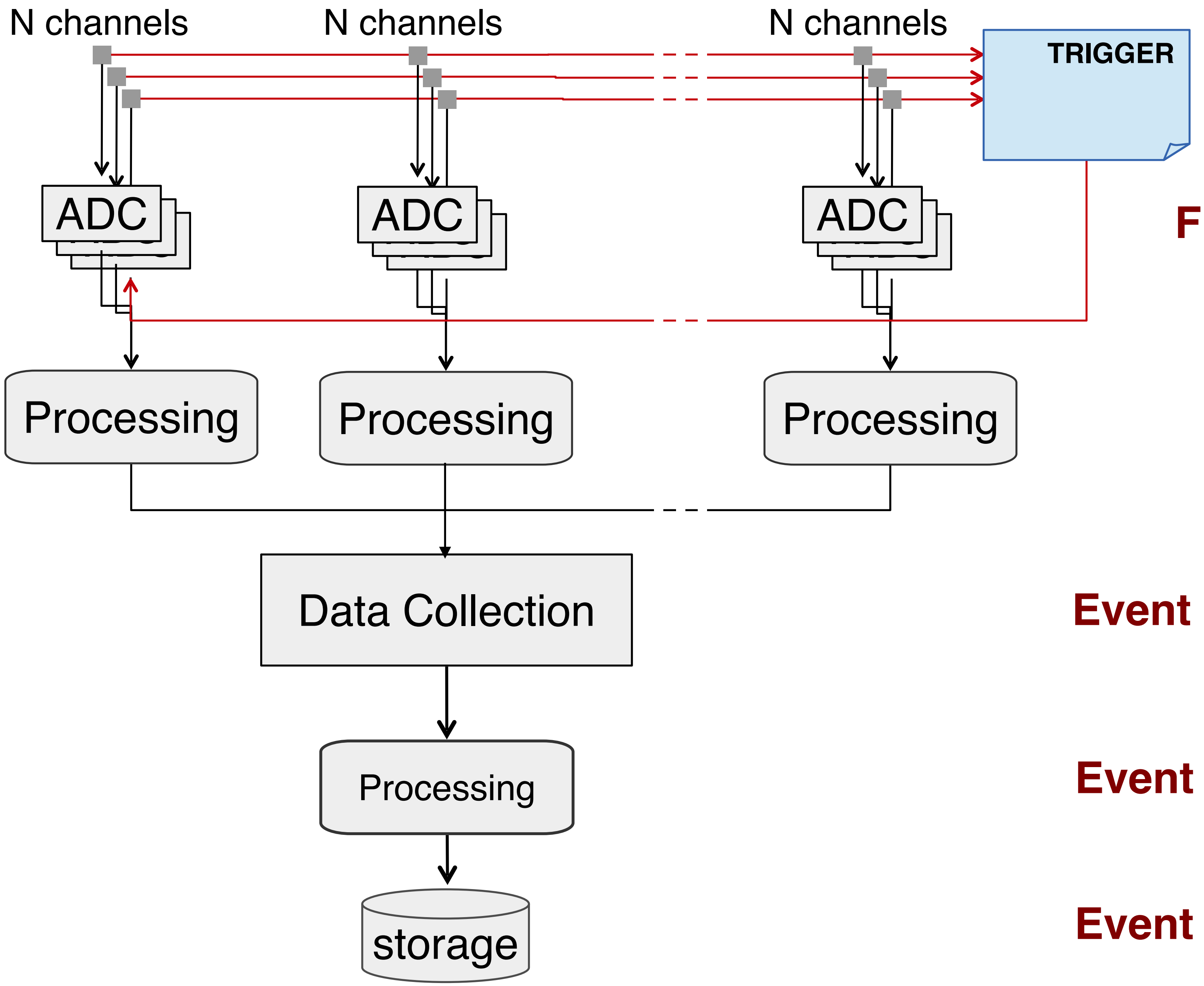
- **Adding more channels requires a hierarchical structure committed to the data handling and conveyance**



Trigger	clock synch trigger decision
Front-End	data digitization data buffering
Readout	data extraction data formatting data buffering
Event Building	event assembly event buffering
Event Filtering	event rejection event buffering
Event Logging	file storage file buffering

ADDING MORE CHANNELS

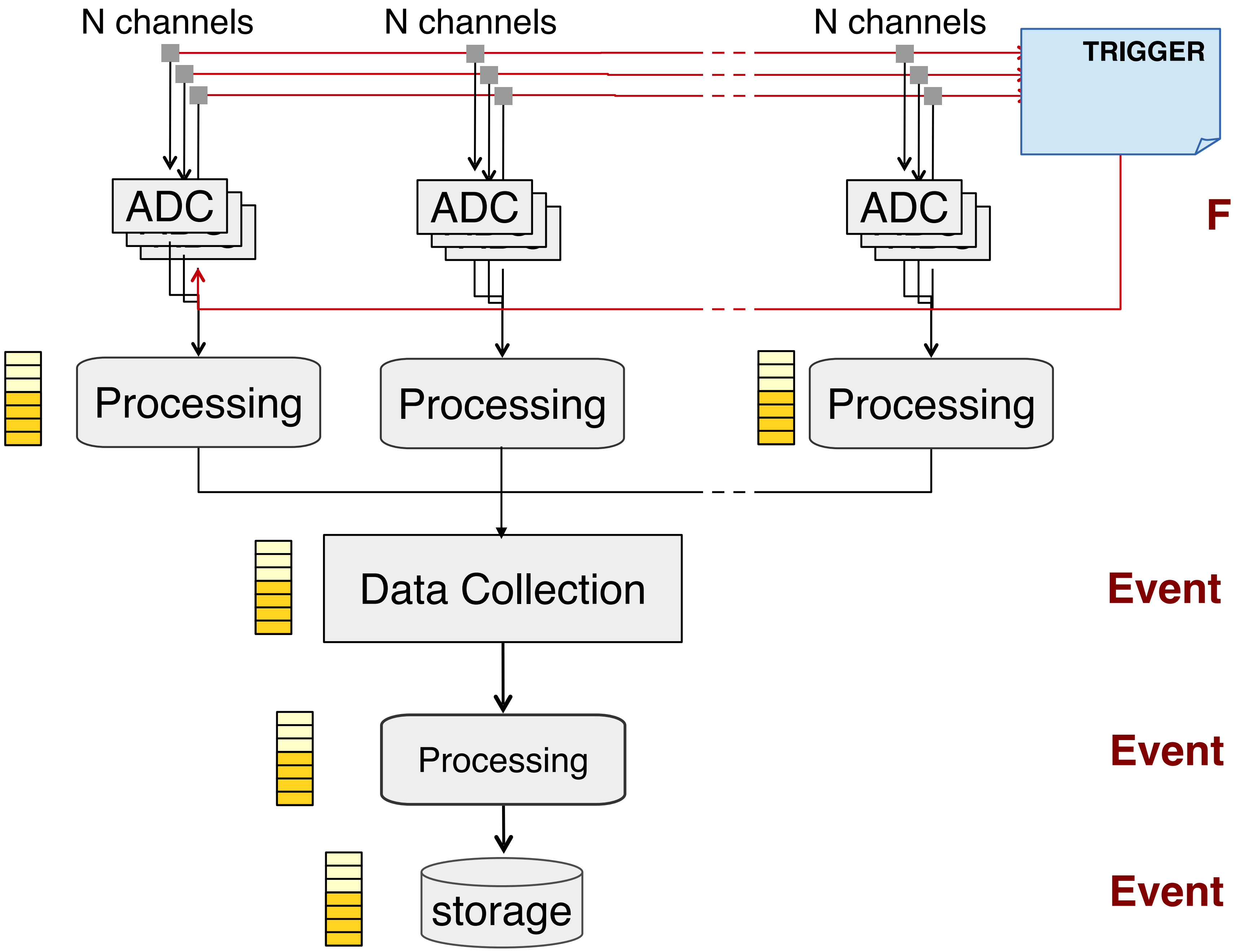
- **Adding more channels requires a hierarchical structure committed to the data handling and conveyance**



Trigger	clock synch trigger decision
Front-End	data digitization data buffering
Readout	data extraction data formatting data buffering
Event Building	event assembly event buffering
Event Filtering	event rejection event buffering
Event Logging	file storage file buffering

ADDING MORE CHANNELS

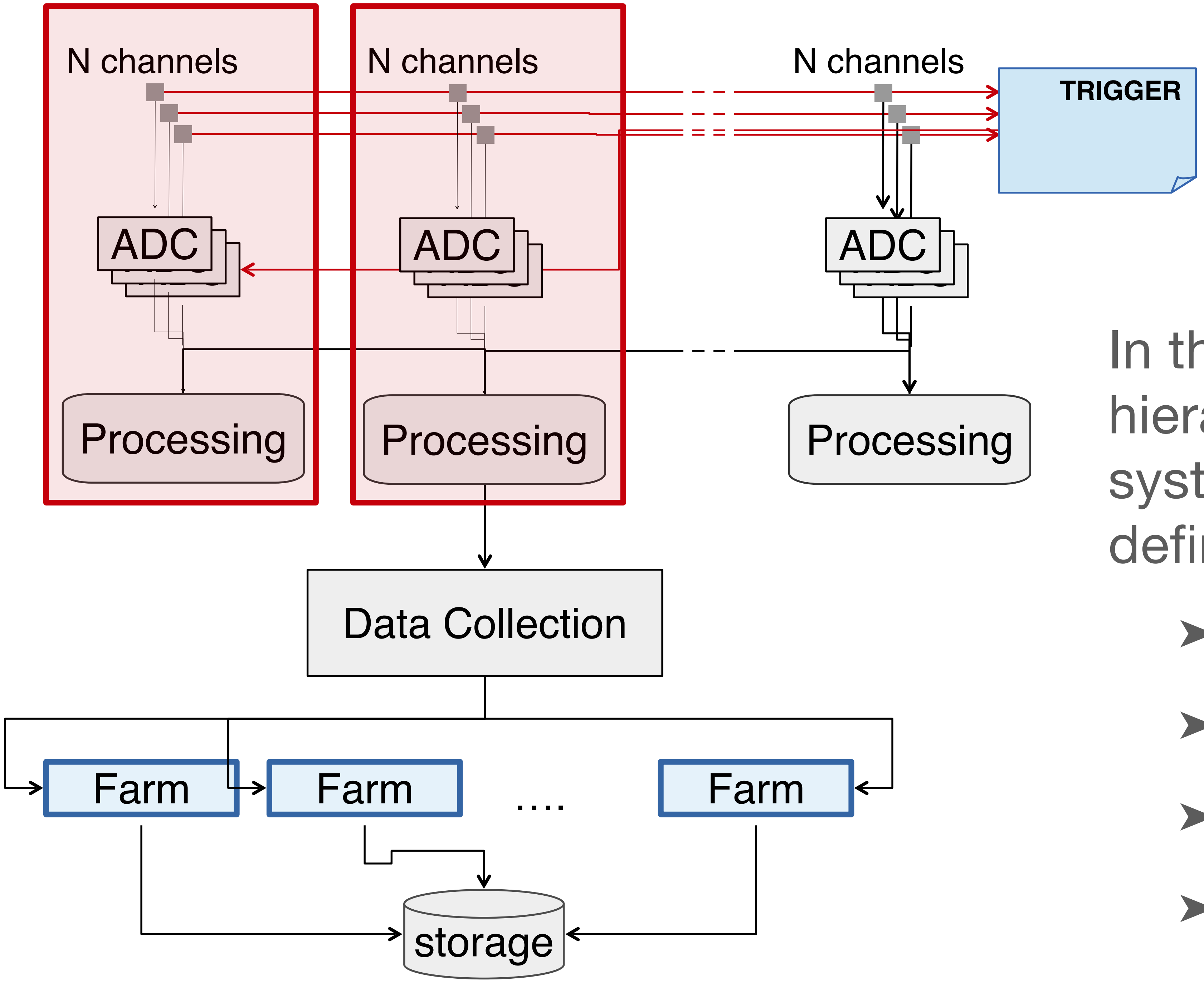
- **Buffering usually needed at every level**
- **DAQ can be seen as a multi level buffering system**



Trigger	clock synch trigger decision
Front-End	data digitization data **buffering**
Readout	data extraction data formatting data **buffering**
Event Building	event assembly event **buffering**
Event Filtering	event rejection event **buffering**
Event Logging	file storage file **buffering**

BUILDING BLOCKS

➡ **Reading out data or building events out of many channels requires many components**

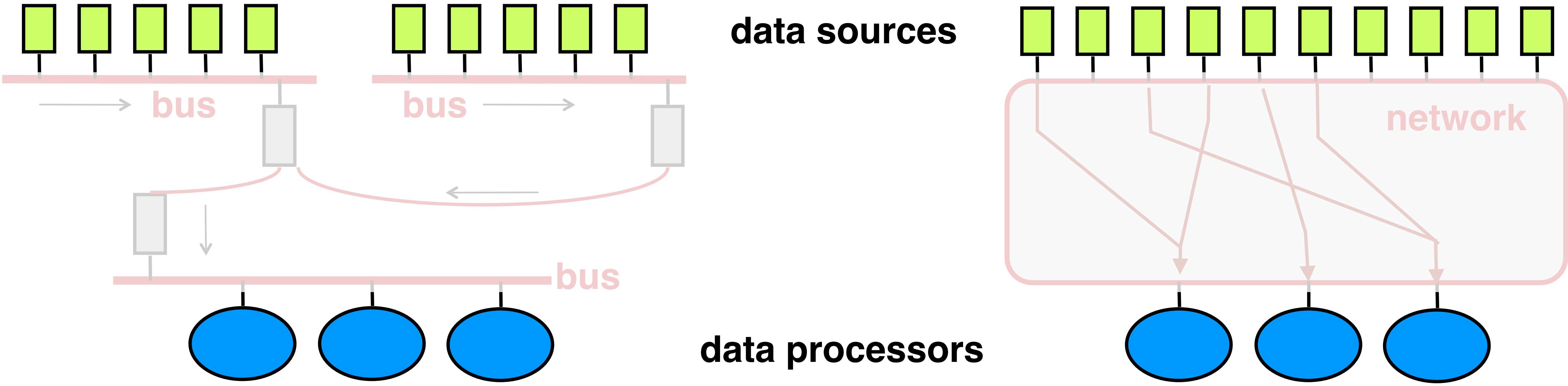


In the design of our hierarchical data-collection system, we have better define “**building blocks**”

- ➤ Readout crates
- ➤ HLT racks
- ➤ Event Building groups
- ➤ DAQ slices

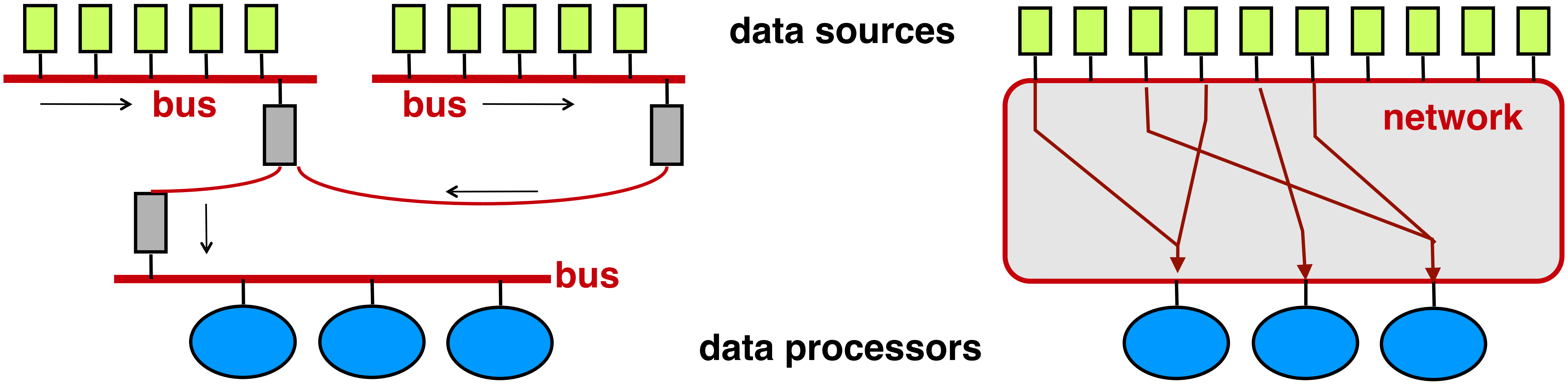
READOUT TOPOLOGY

- ➡ **How to organize the interconnections inside the building blocks and between building blocks?**
  - ➡ How to connect data sources and data destinations?
  - ➡ Two main classes: **bus** or **network**



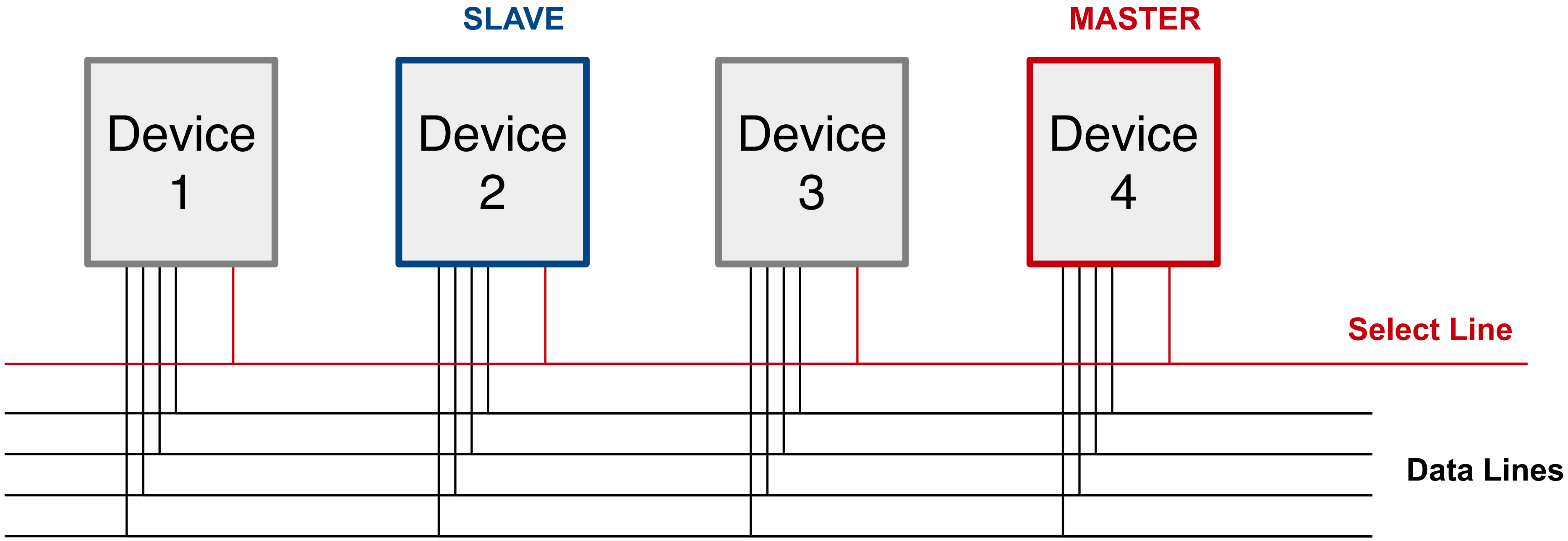
READOUT TOPOLOGY

- **How to organize the interconnections inside the building blocks and between building blocks?**
  - How to connect data sources and data destinations?
  - Two main classes: **bus** or **network**

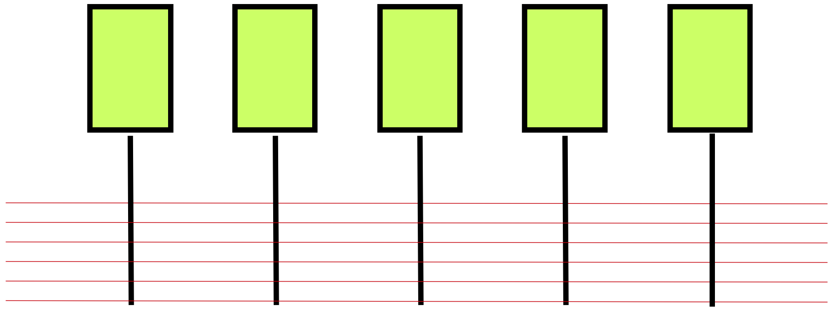


BUSES

- ➡ **Devices connected via a shared bus**
  - ➡ Bus → group of electrical lines
- ➡ **Sharing implies arbitration**
  - ➡ Devices can be **master** or **slave**
  - ➡ Devices can be addresses (uniquely identified) on the bus
- ➡ **E.g.: SCSI, Parallel ATA, VME, PCI …**
  - ➡ local, external, crate, long distance, ...

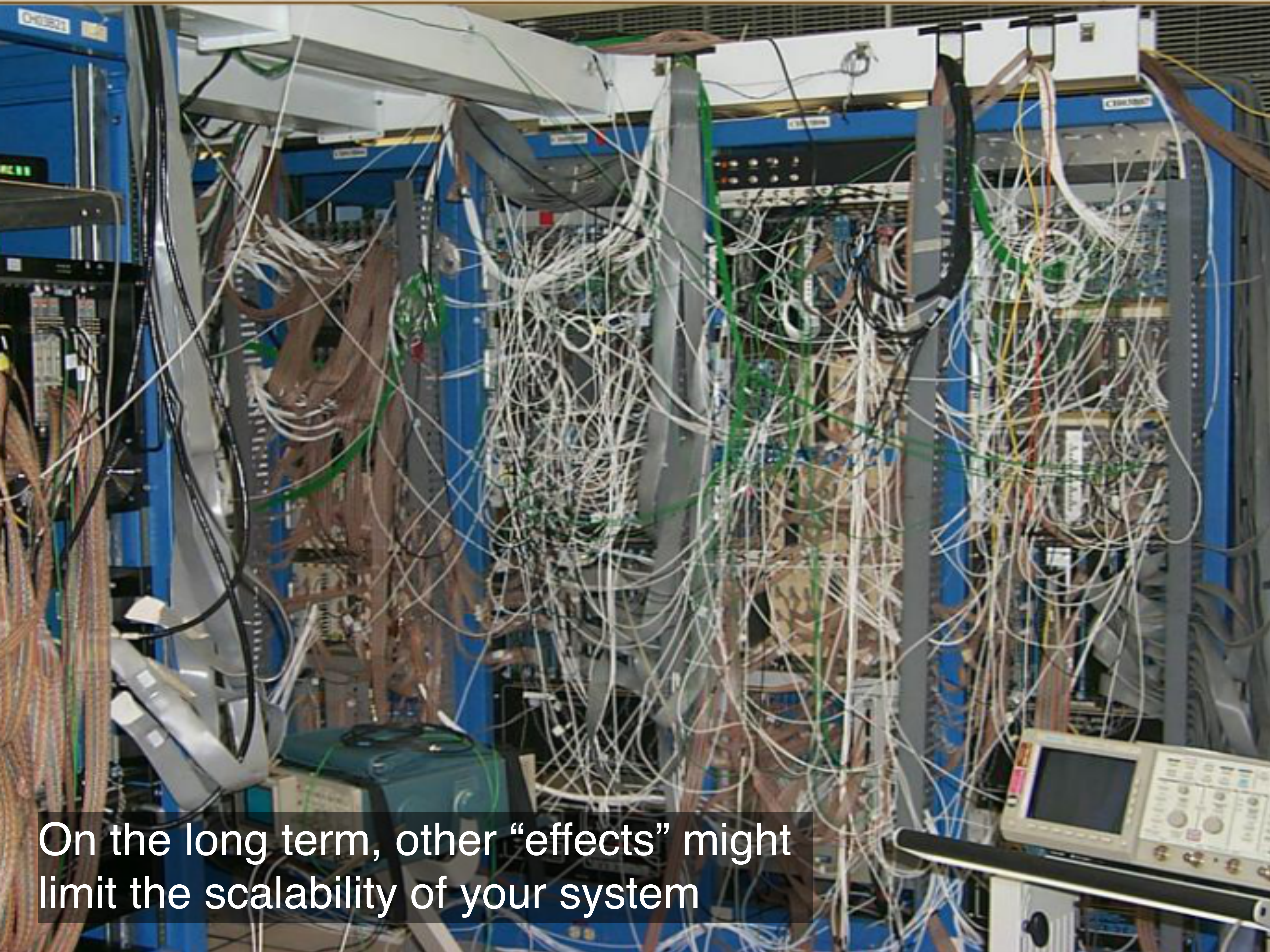


BUS FACTS



- **Simple** :-)
  - Fixed number of lines (bus-width)
  - Devices have to follow well defined interfaces
    - Mechanical, electrical, communication, ...
- **Scalability issues** :-(
  - Bus bandwidth is shared among all the devices
  - Maximum bus width is limited
  - Maximum number of devices depends on bus length
  - Maximum bus frequency is inversely proportional to the bus length

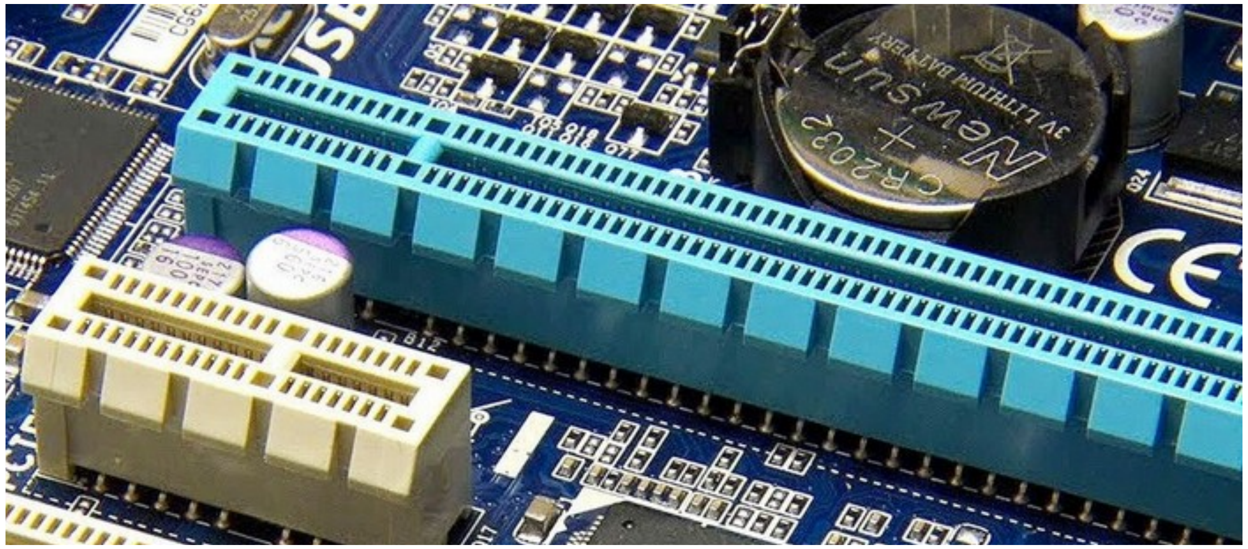
  - **On the long term, other “effects” might limit the scalability of your system**



On the long term, other “effects” might limit the scalability of your system

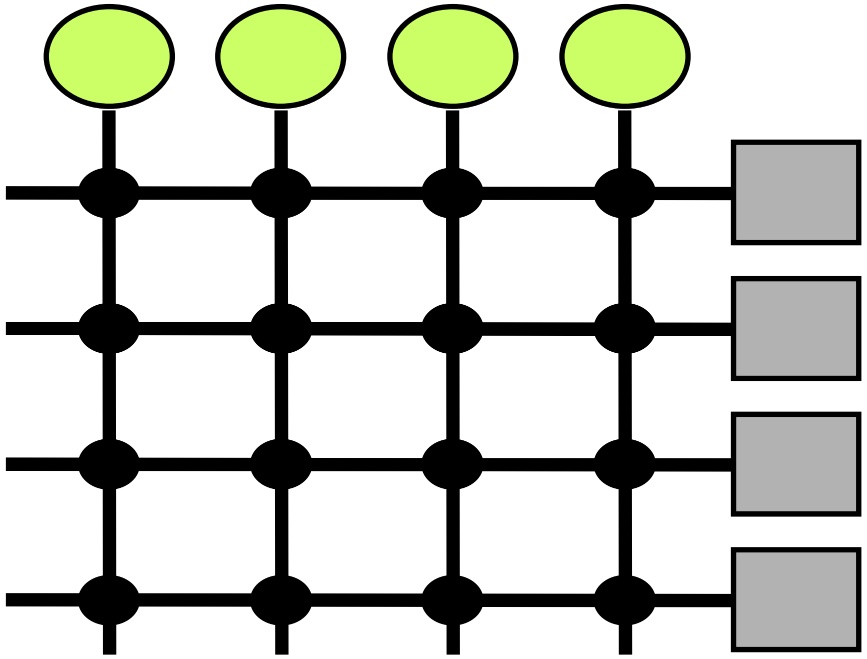
BUS STANDARDS

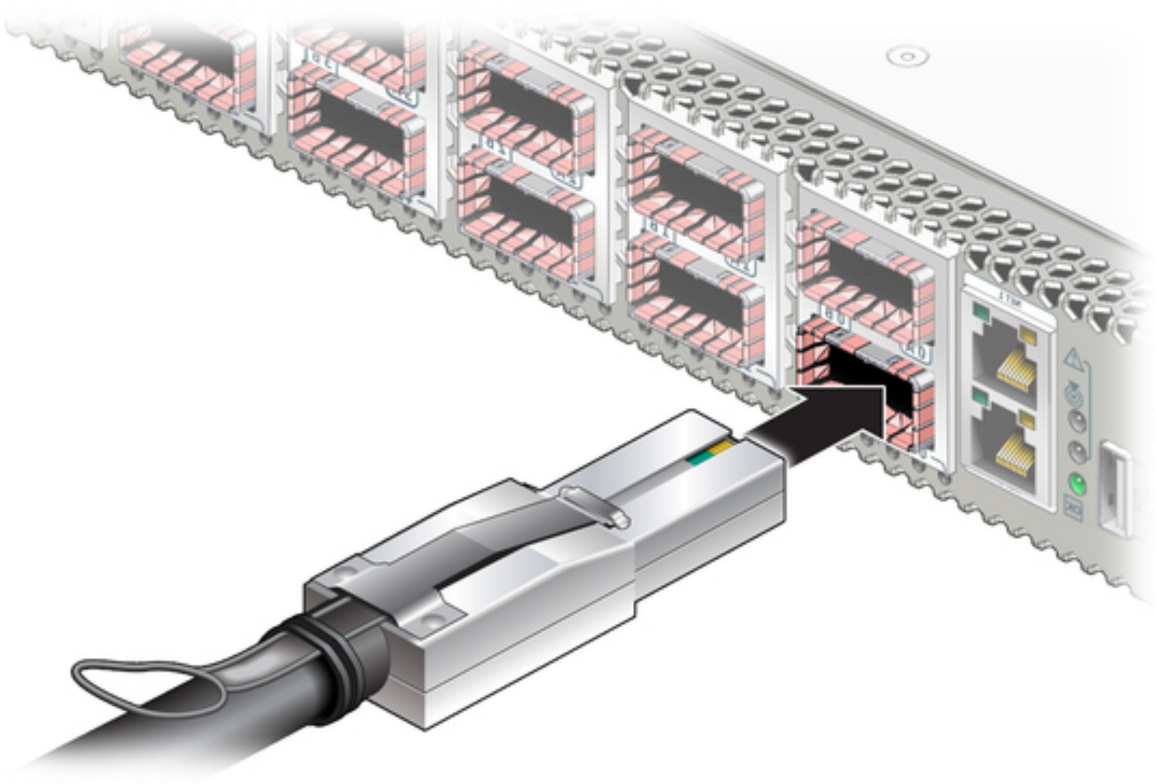
- ➡ VME Modular electronics
- ➡ VME bus programming
- ➡ µATCA
- ➡ PCI express



NETWORK

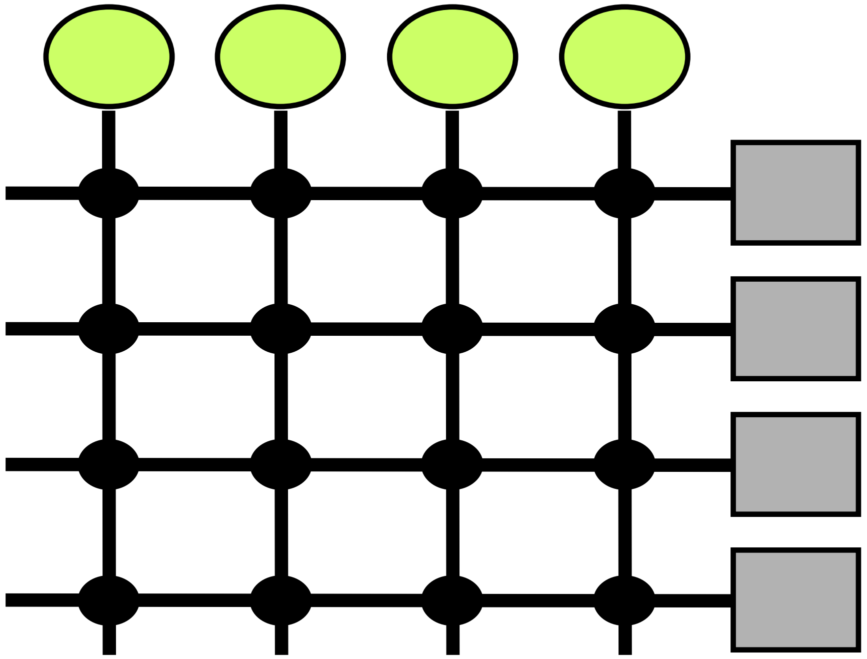
- ➡ **All devices are equal**
  - ➡ They communicate directly with each other via messages
  - ➡ No arbitration, simultaneous communications
- ➡ **Eg: Telephone, Ethernet, Infiniband, …**

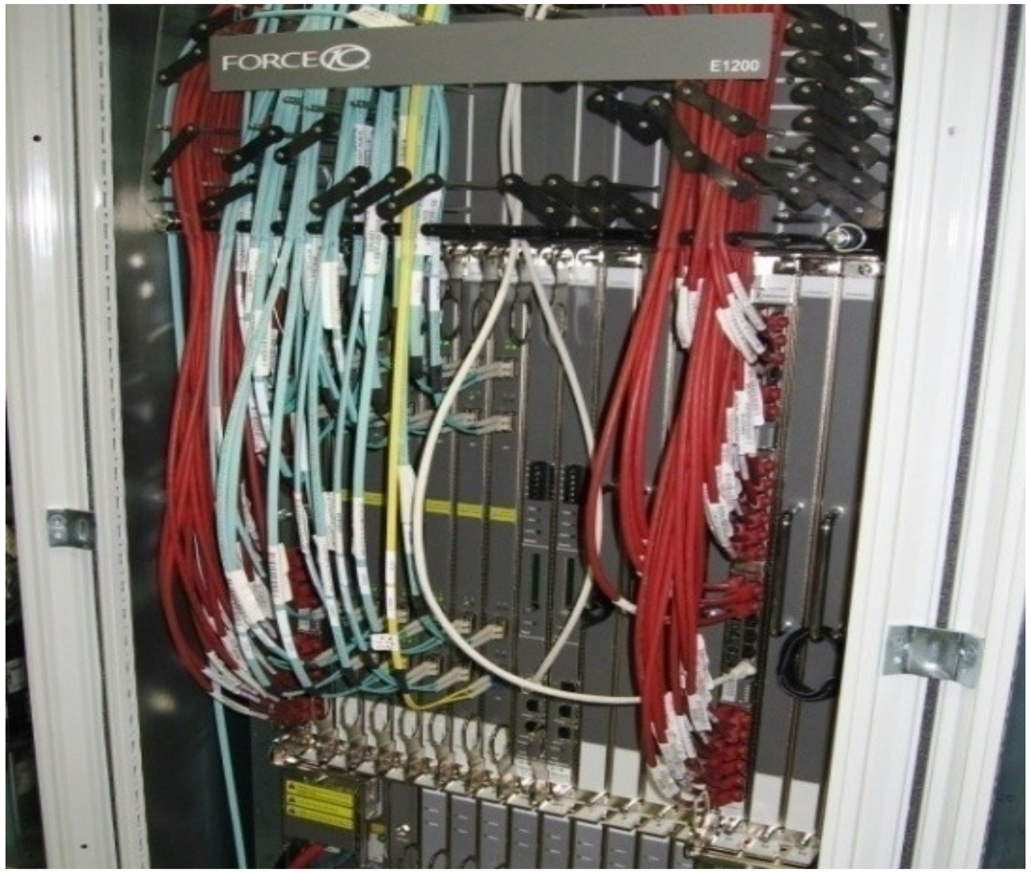




NETWORK

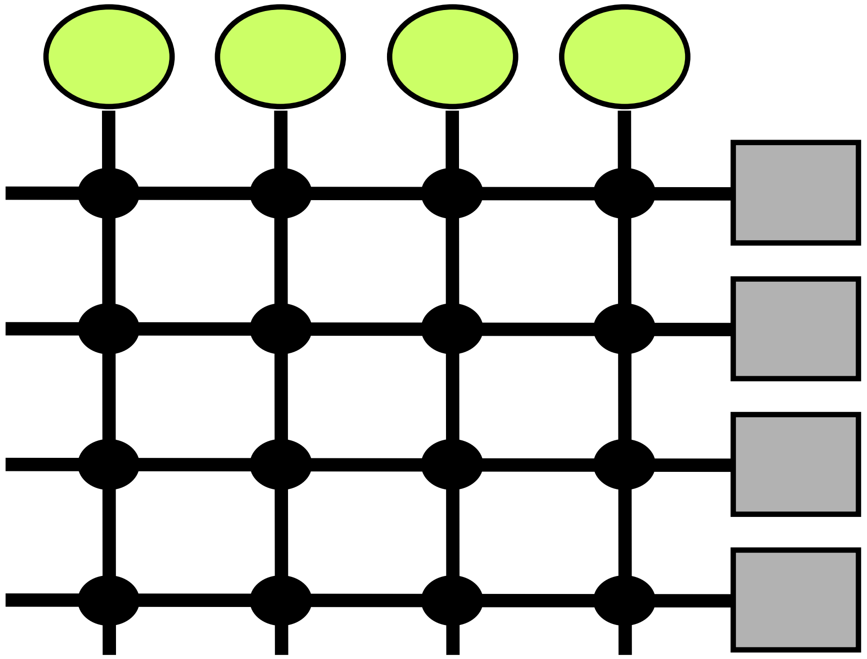
- In switched networks, **switches** move messages between sources and destinations
  - Find the right path
- How **congestions** (two messages with the same destination at the same time) are handled?
  - The key is .... buffering

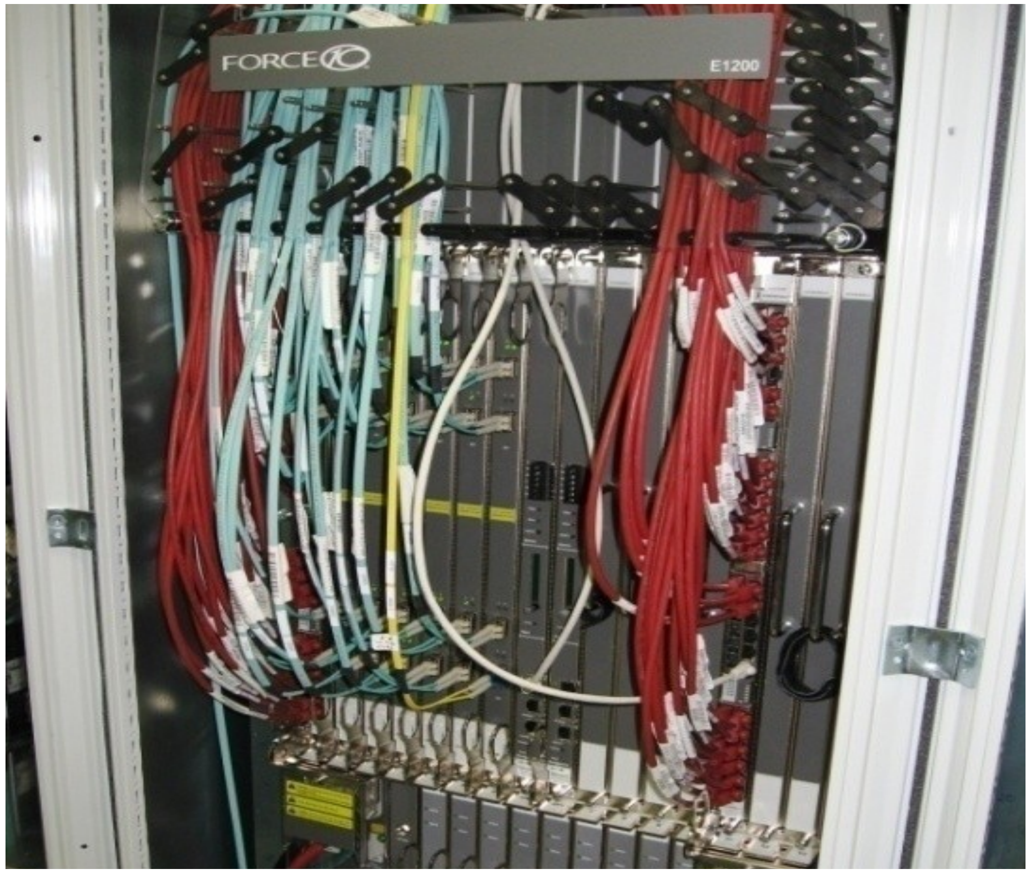




NETWORK

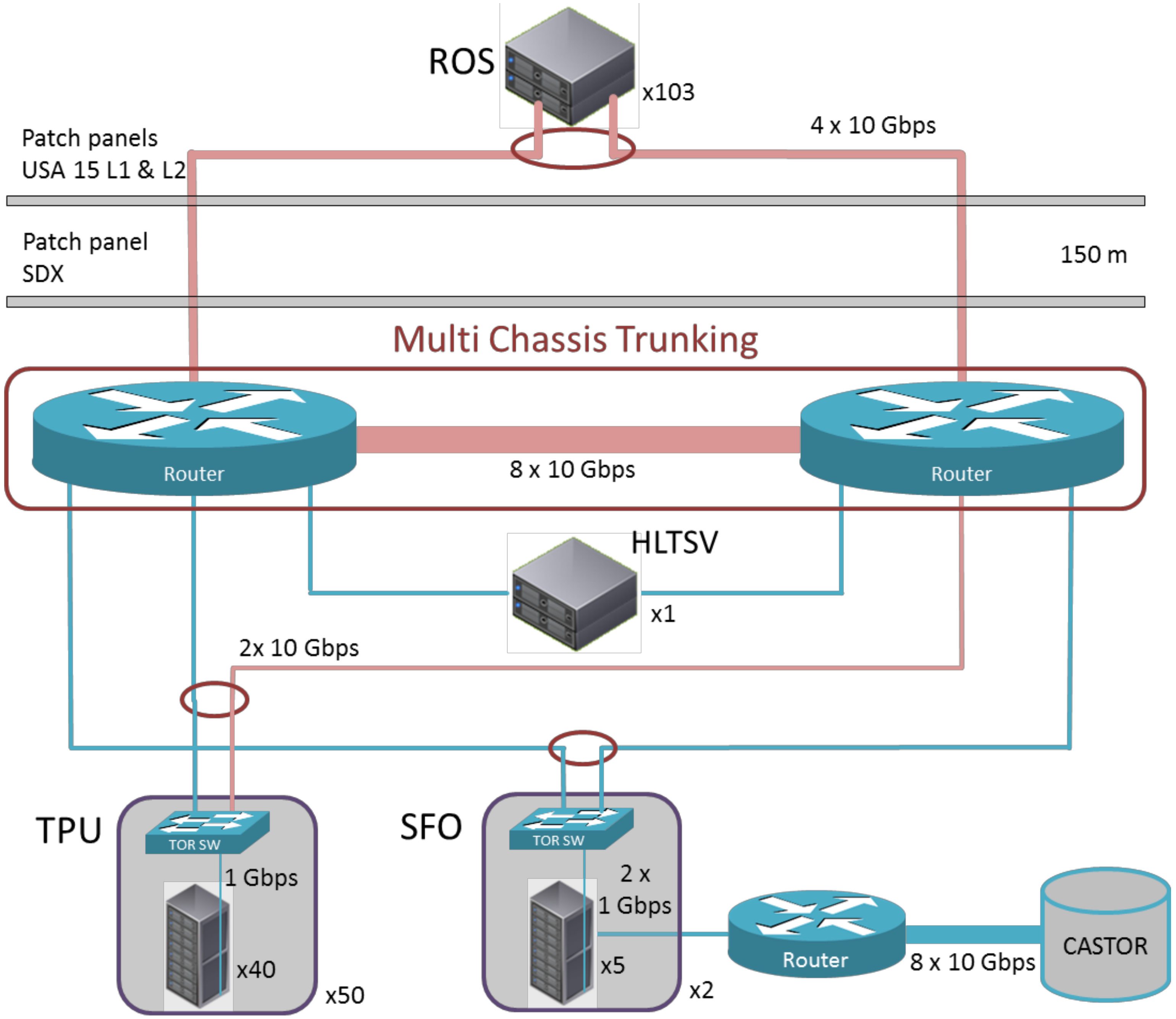
- In switched networks, **switches** move messages between sources and destinations
  - Find the right path
- How **congestions** (two messages with the same destination at the same time) are handled?
  - The key is .... **buffering**





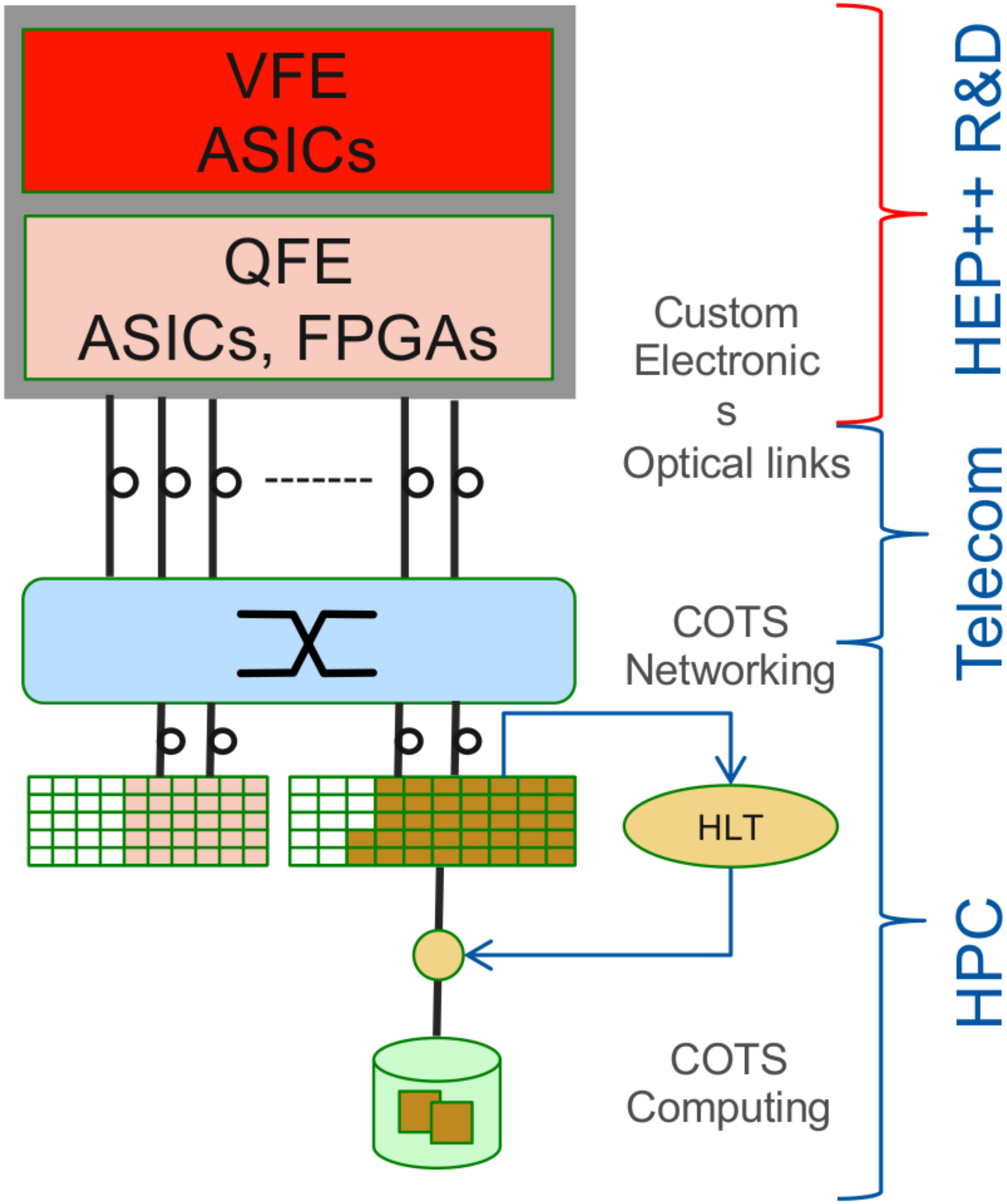
NETWORK

- ➡ **Networks scale well (and allow redundancy)**
  - ➡ They are the backbones of the LHC DAQ systems



RECAP

- **Very Front End**
  - does the analog part
  - ADC, low-level calibration, zero suppression, lossless compression
  - low-power, rad-tolerant
- **Quasi Front End**
  - medium scale aggregation, local reconstruction, ”lossy” compression, transition to standard protocol on optical links
- **Commodity Of The Shelf**
  - COTS switched networks
  - COTS servers, with co-processors (GPU, FPGA)

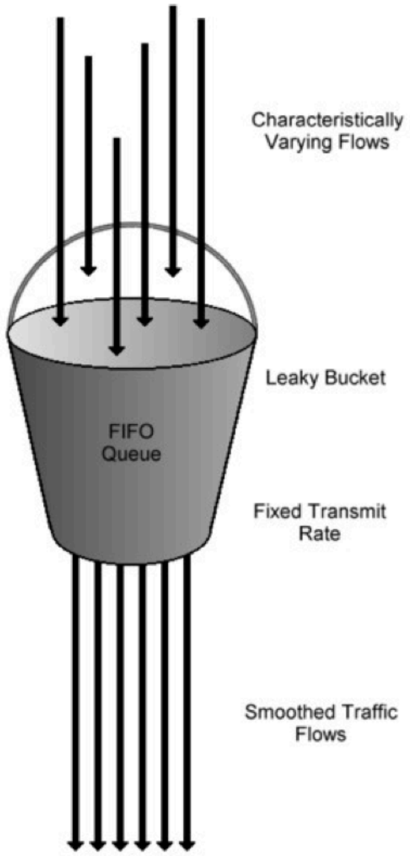


E. Meschi, Summer Student Lectures 2022

FIGHTING BOTTLENECKS

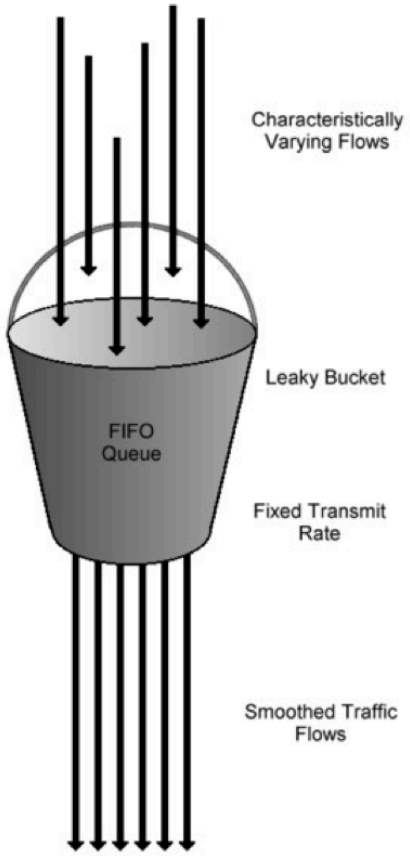
- ➡ Artificial deadtime
- ➡ Data collection
- ➡ Multi-level trigger
- ➡ Data-flow control
- ➡ Data reconstruction

1 – ARTIFICIAL DEADTIME



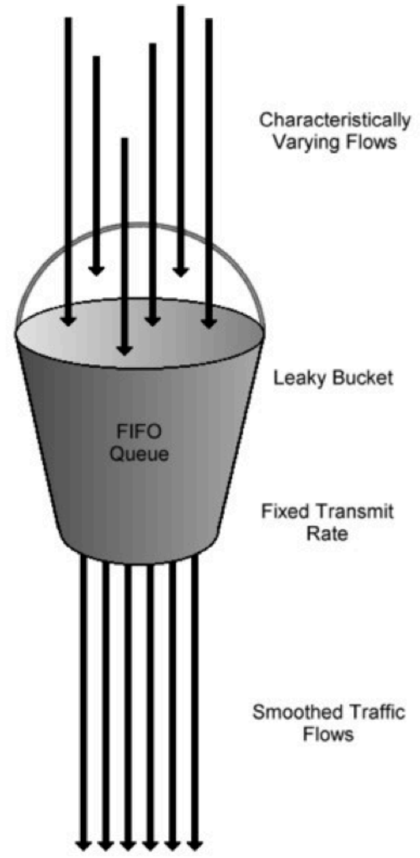
1 – ARTIFICIAL DEADTIME

- **If two signals arrive very close in time**
  - detector signals overlap (ask you detector expert, are you sure the detector is good at that rate? is your FE fast enough?)
  - can have dead-time if not added any … FIFO!



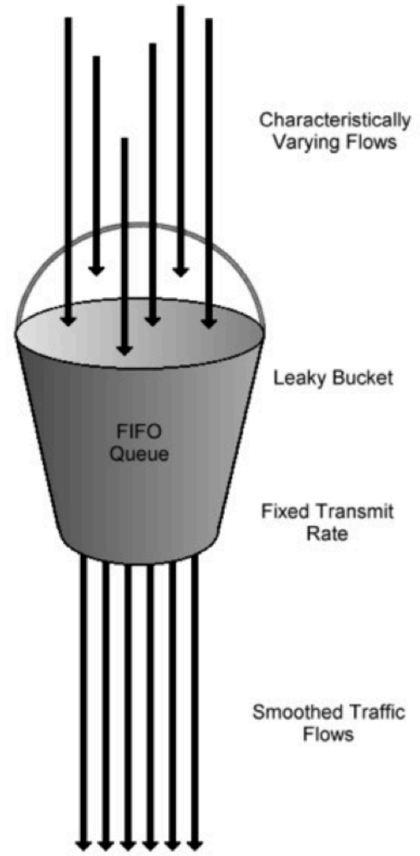
1 – ARTIFICIAL DEADTIME

- **If two signals arrive very close in time**
  - detector signals overlap (ask you detector expert, are you sure the detector is good at that rate? is your FE fast enough?)
  - can have dead-time if not added any … FIFO!
- **Is derandomization enough?**
  - if FE readout windows overlap
    - add artificial dead-time to protect the FrontEnd (**simple deadtime**)
  - if FE buffers overflow in case of trigger bursts
    - add artificial dead-time (**complex deadtime**)



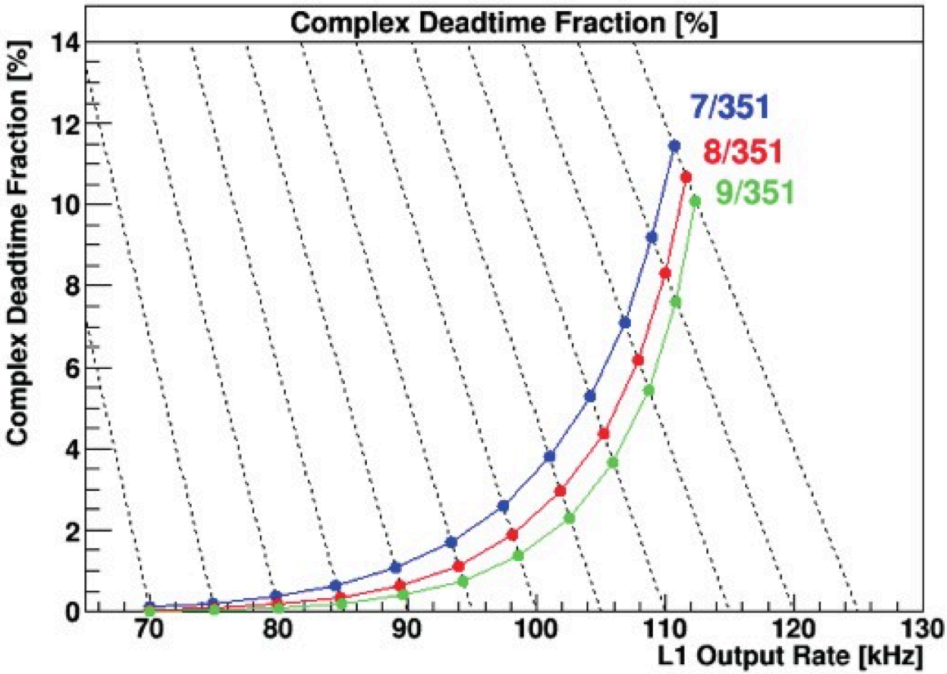
1 - ARTIFICIAL DEADTIME

- **If two signals arrive very close in time**
  - detector signals overlap (ask you detector expert, are you sure the detector is good at that rate? is your FE fast enough?)
  - can have dead-time if not added any … FIFO!
- **Is derandomization enough?**
  - if FE readout windows overlap
    - add artificial dead-time to protect the FrontEnd (**simple deadtime**)
  - if FE buffers overflow in case of trigger bursts
    - add artificial dead-time (**complex deadtime**)

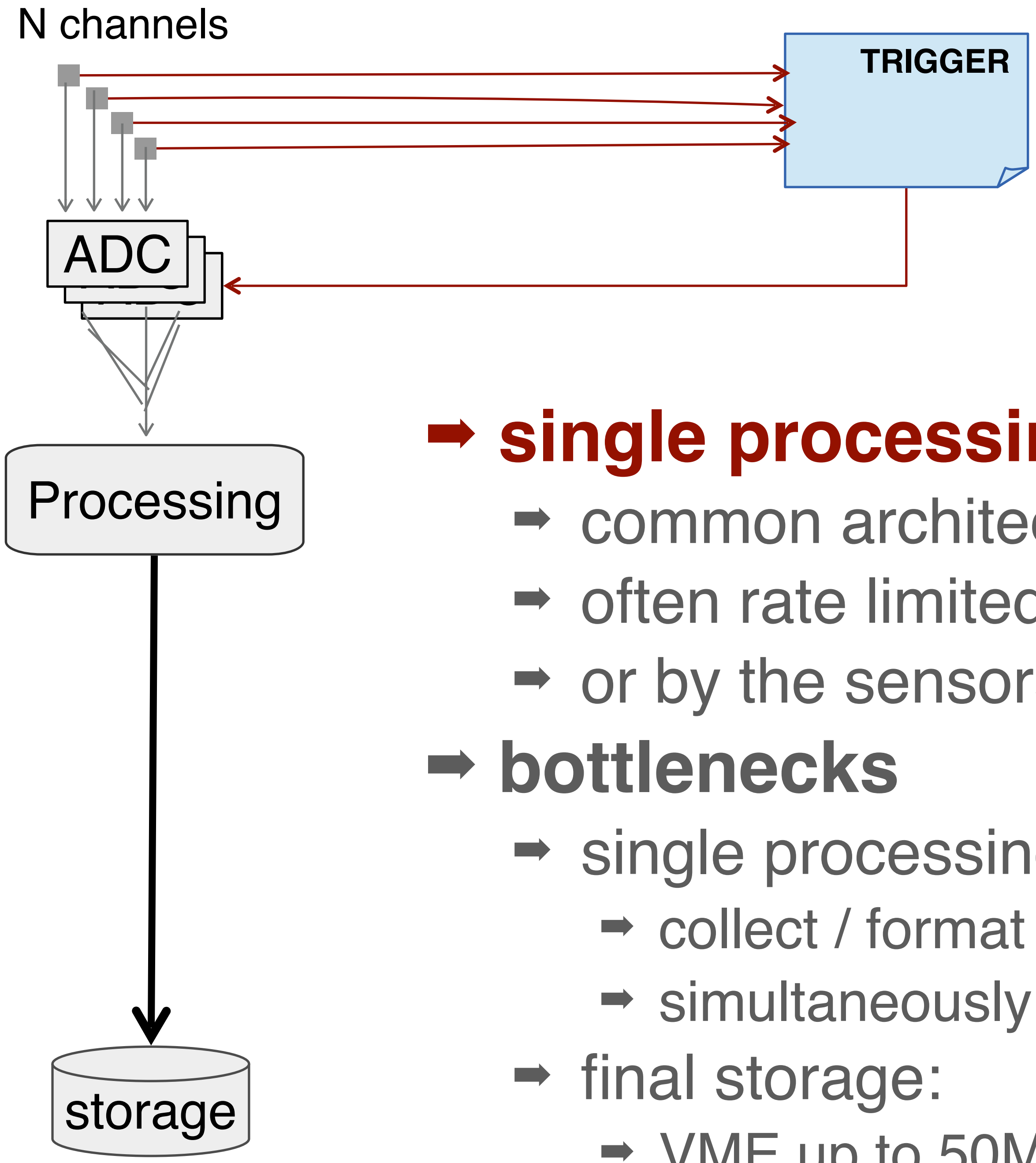


- **Example in ATLAS @Run2: 90 kHz, < 2%**
  - Simple deadtime: 4 LHC BC [100 ns] after any L1 trigger
  - Complex deadtime: leaky-bucket algorithms x4 detectors
    - two parameters: bucket size (in number of events) / readout time (in BC units)
    - i.e. 9 / 351 for LAr readout

Leaky bucket (LAr readout)



2 - DATA COLLECTION

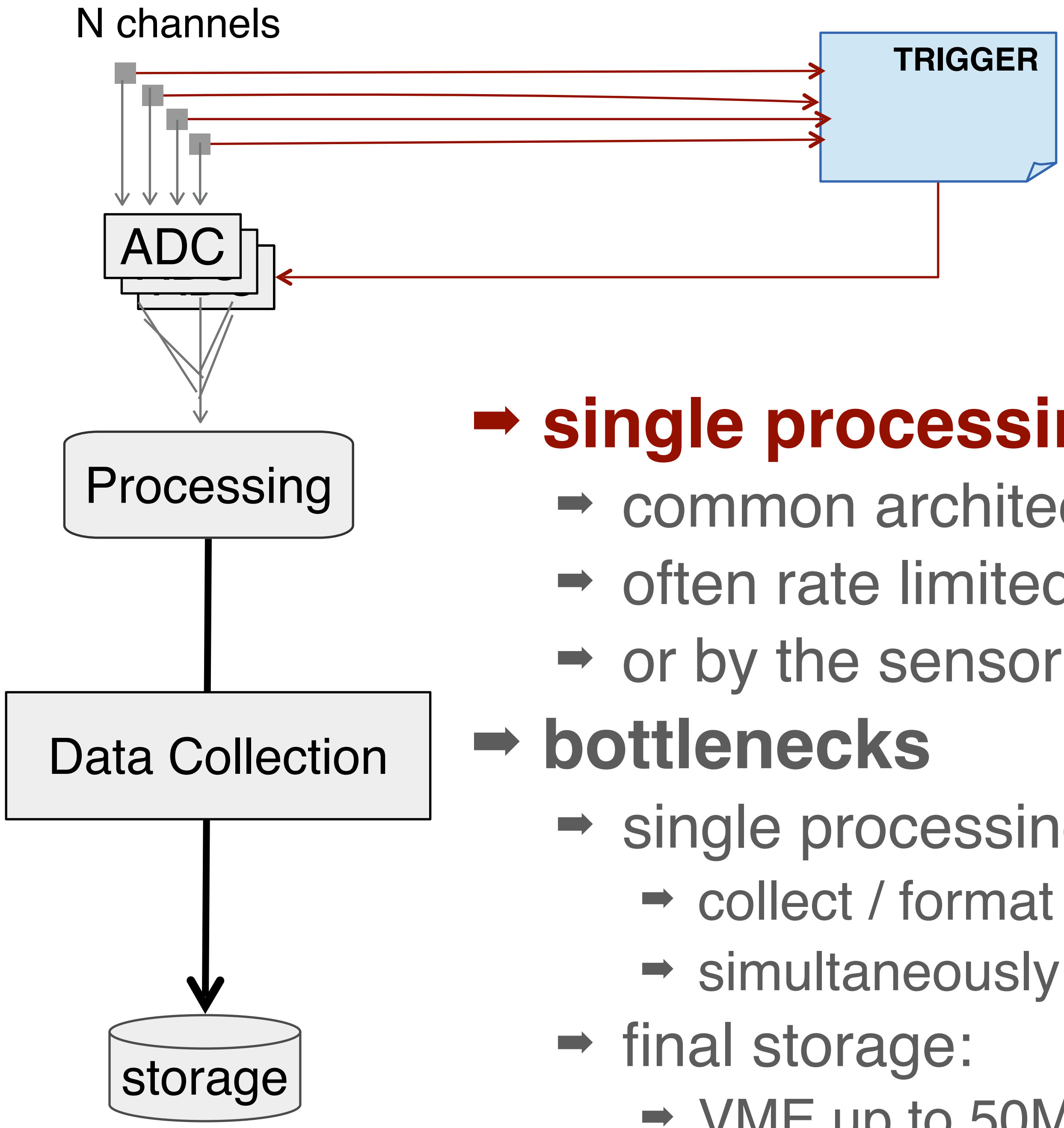


more sensors ⇒ more granularity

multiple digitisers ⇒ more parallelism

- ➡ **single processing system**
  - ➡ common architecture in **test-beams and small experiments**
  - ➡ often rate limited by (interesting) physics itself, not TDAQ
  - ➡ or by the sensors
- ➡ **bottlenecks**
  - ➡ single processing unit
    - ➡ collect / format / compress data can be heavy
    - ➡ simultaneously writing to storage
  - ➡ final storage:
    - ➡ VME up to 50MB/s → 1TB in 6h
    - ➡ too many disks in one week!
- ➡ **Data Collection unit decouples storage from processing**
  - ➡ dedicated to format, compress and store

2 – DATA COLLECTION



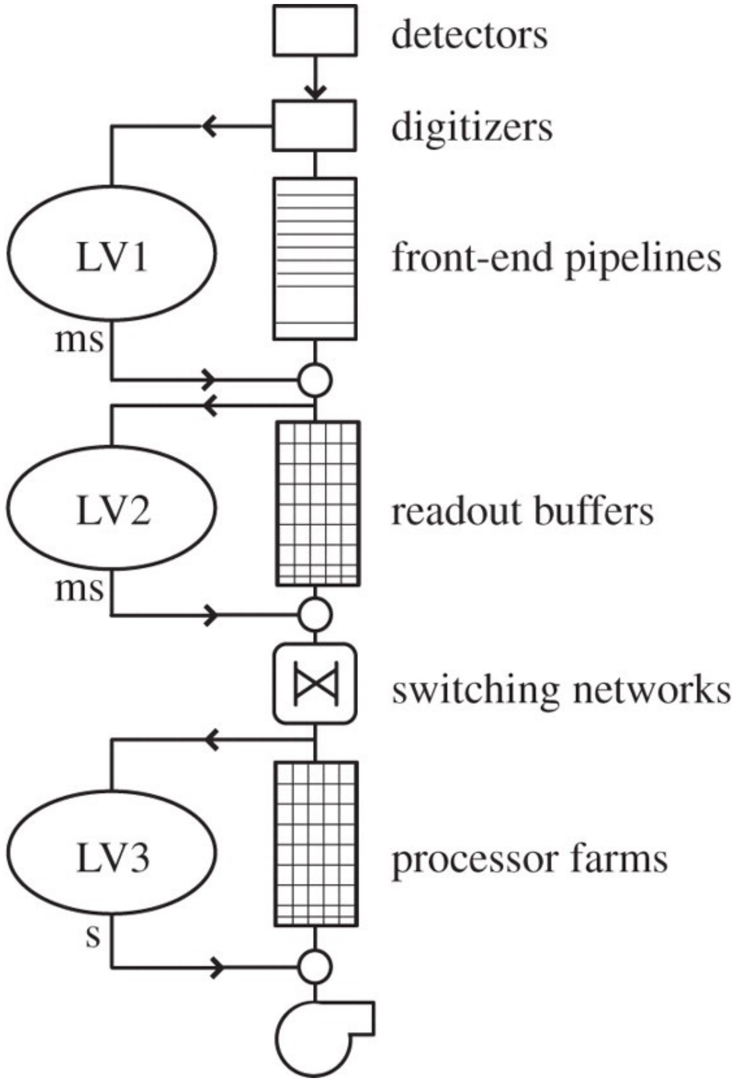
more sensors ⇒ more granularity

multiple digitisers ⇒ more parallelism

- ➡ **single processing system**
  - ➡ common architecture in **test-beams and small experiments**
  - ➡ often rate limited by (interesting) physics itself, not TDAQ
  - ➡ or by the sensors
- ➡ **bottlenecks**
  - ➡ single processing unit
    - ➡ collect / format / compress data can be heavy
    - ➡ simultaneously writing to storage
  - ➡ final storage:
    - ➡ VME up to 50MB/s → 1TB in 6h
    - ➡ too many disks in one week!
- ➡ **Data Collection unit decouples storage from processing**
  - ➡ dedicated to format, compress and store

3 - MULTI-LEVEL TRIGGER

- ➡ **Reduce the rate at each stage, with limited buffer size and no deadtime**
  - ➡ $\tau \sim \lambda$ (traffic intensity~1)
- ➡ **High level triggers with longer latency**
  - ➡ more complex filters
  - ➡ more data (for example silicon detectors)



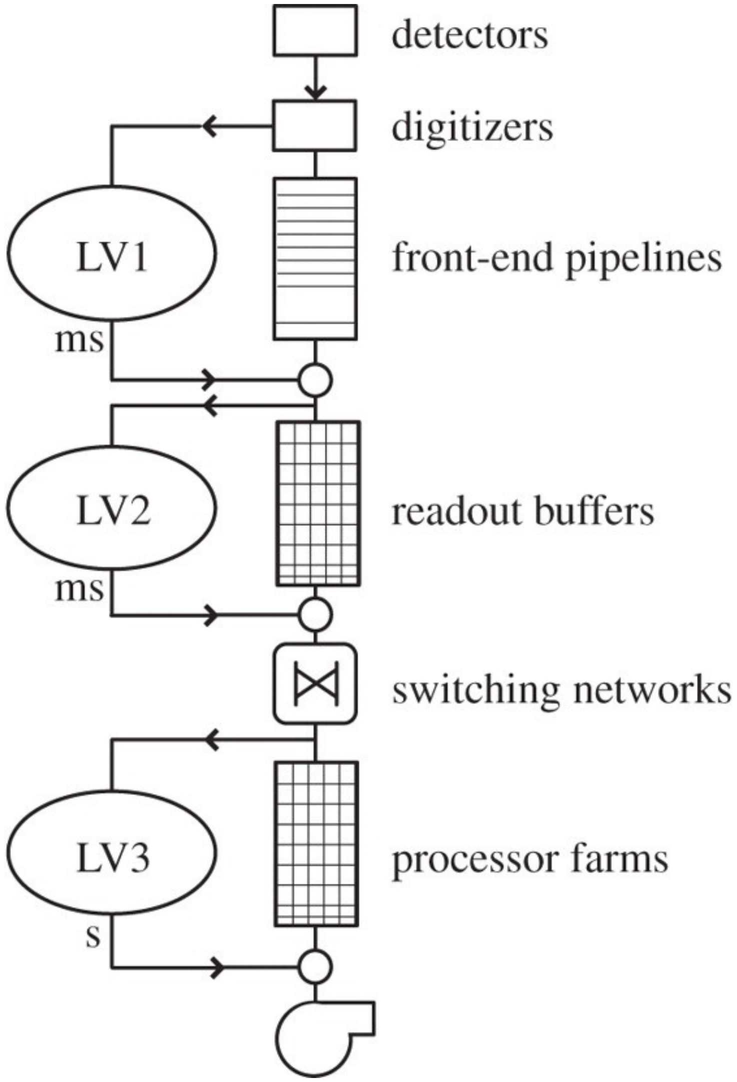
Recall on trigger architectures

- ➡ **Real time system**
  - ➡ must respond within some **fixed latency**
  - ➡ Latency = max Latency
  - ➡ over fluctuations is bad, will create deadtime
- ➡ **Non-real-time system**
  - ➡ responds as soon as it's available
  - ➡ Latency = **Mean Latency**
  - ➡ over fluctuations is fine, shouldn't create deadtime

CERN - LEP

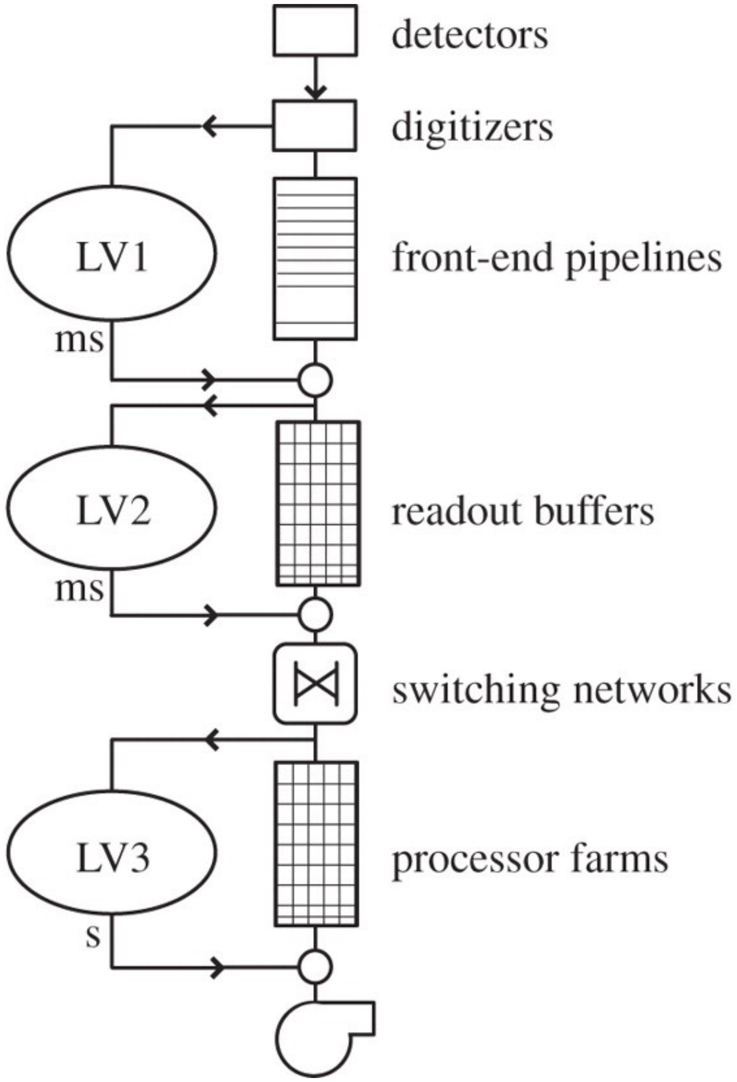
- ➡ 10^5 channels
- ➡ 22 µs crossing – no event overlap
- ➡ single interaction
- ➡ L1 ~ 10^3 Hz
- ➡ L2 ~ 10^2 Hz
- ➡ L3 ~ 10 Hz
- ➡ 100 kB/ev → 1 MB/s

4 – DATAFLOW CONTROL



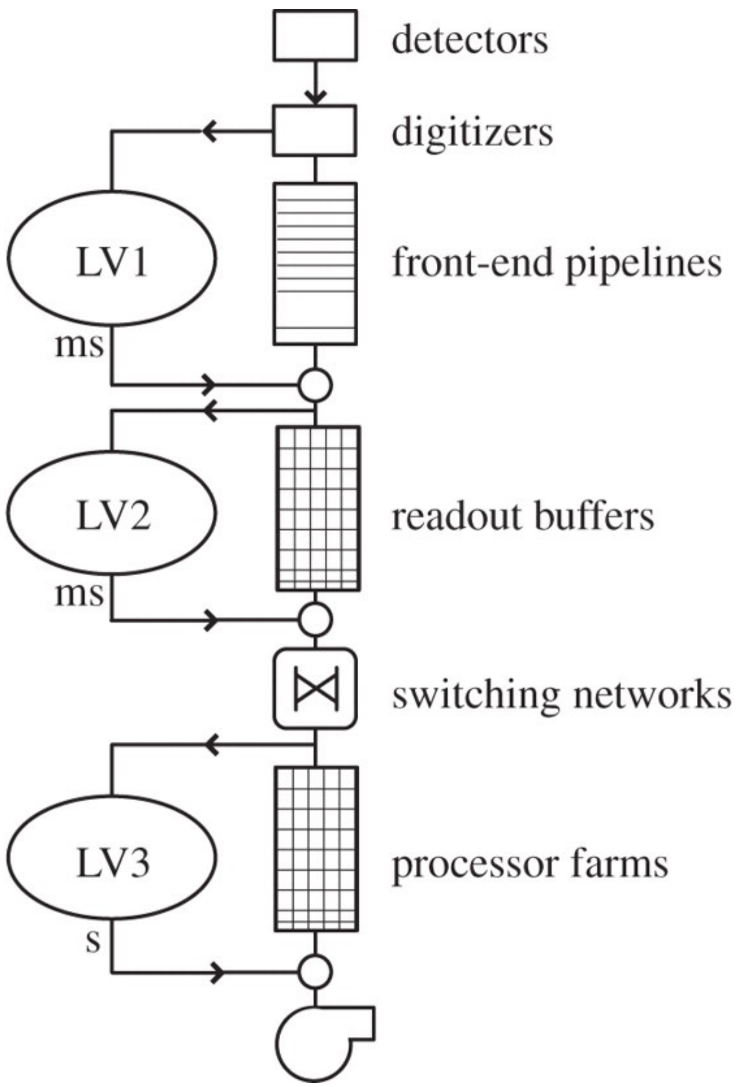
4 – DATAFLOW CONTROL

- **Buffers are not the "final solution"**
  - Can overflow, with bursts and unusual event sizes
  - In these cases, can
    - discard data locally or
    - exert "**back-pressure**", i. e. ask previous level(s) to block the dataflow



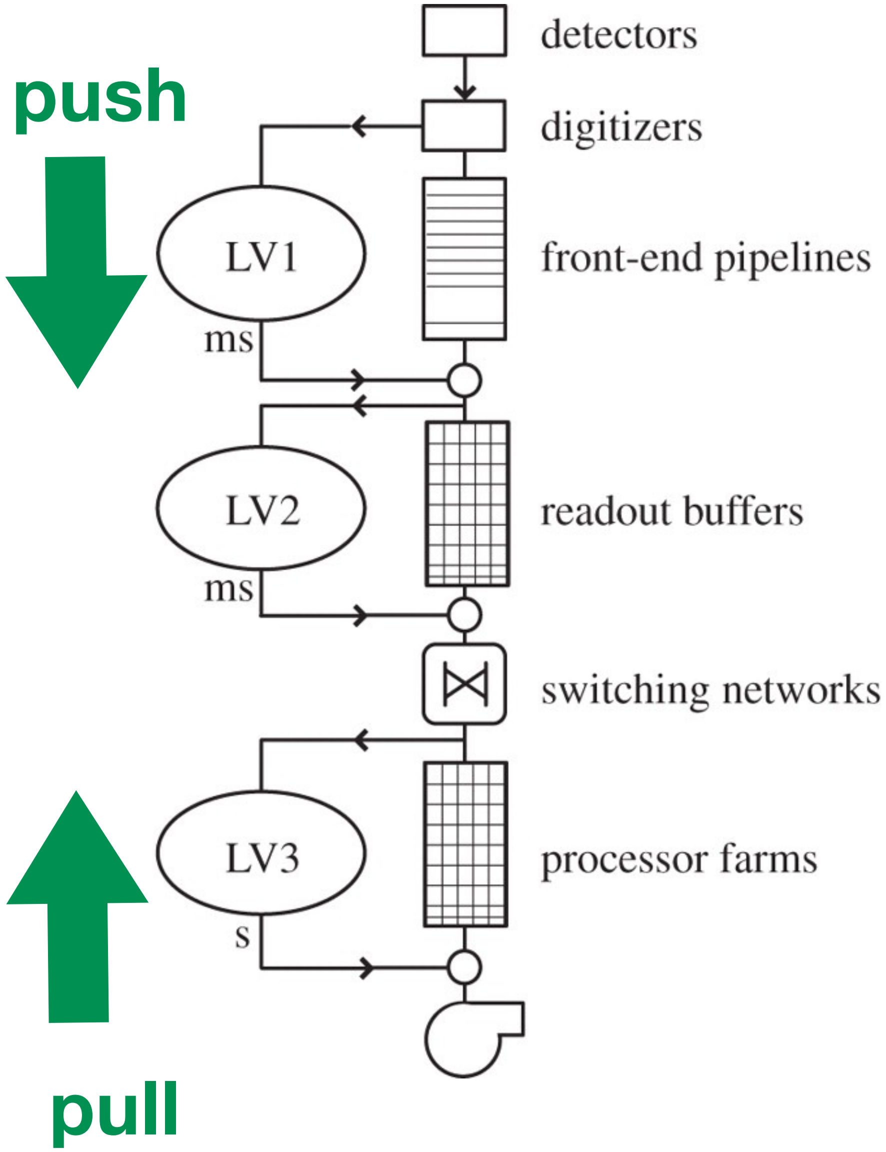
4 – DATAFLOW CONTROL

- **Buffers are not the "final solution"**
  - Can overflow, with bursts and unusual event sizes
  - In these cases, can
    - discard data locally or
    - exert "**back-pressure**", i. e. ask previous level(s) to block the dataflow
- **Throughput optimization means avoiding dead-time due to back-pressure**
  - using knowledge of the input buffer state



4 – DATAFLOW CONTROL

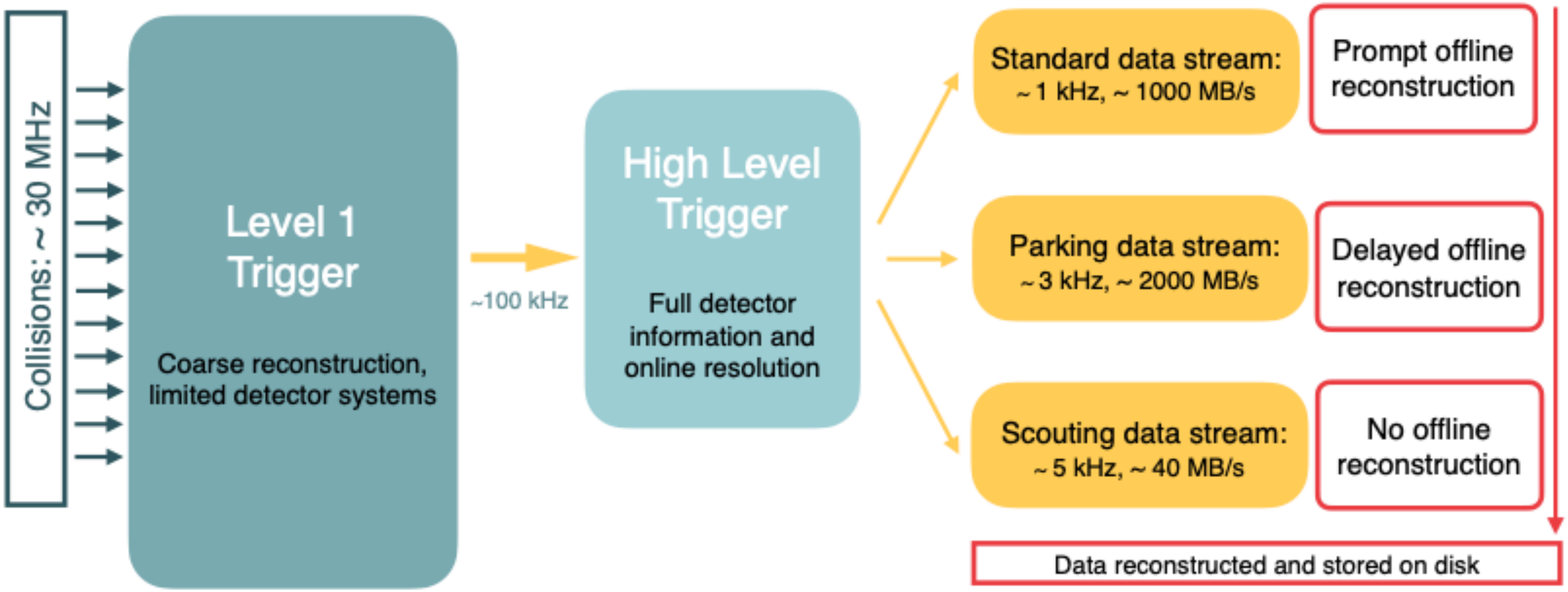
- **Buffers are not the "final solution"**
  - Can overflow, with bursts and unusual event sizes
  - In these cases, can
    - discard data locally or
    - exert "**back-pressure**", i. e. ask previous level(s) to block the dataflow
- **Throughput optimization means avoiding dead-time due to back-pressure**
  - using knowledge of the input buffer state



- **Who controls the flow? FE (push) or EB (pull)**
  - **FE Push**: Events are sent as soon as data are available to the sender (e.g. round-robin algorithm) ⟹ Busy or Throttle (block trigger)
  - **EB Pull** : events are required by a given destination processes (may need an event manager) ⟹ back-pressure (block dataflow)
  - **Push-Pull** ⟹ busy and back-pressure

5 - MOVE FINAL RECONSTRUCTION

- **Can play with data size and delayed reconstruction to overcome limitations**
- **Trigger Level Analysis / data scouting: data compressed via full event reconstruction, avoid to save raw detector data**
  - if the bandwidth to write to the permanent storage is limited
- **Data parking: data saved on temporary storage and reconstructed when resources are available (during fills,….)**
  - if the resources to promptly reconstruct the data in the computing center are limited


Collisions: ~ 30 MHz
Level 1 Trigger
Coarse reconstruction, limited detector systems
~100 kHz
High Level Trigger
Full detector information and online resolution
Standard data stream: ~1 kHz, ~1000 MB/s
Prompt offline reconstruction
Parking data stream: ~3 kHz, ~2000 MB/s
Delayed offline reconstruction
Scouting data stream: ~5 kHz, ~40 MB/s
No offline reconstruction
Data reconstructed and stored on disk

CMS paper, 2024

CONCLUDE WITH GENERAL T/DAQ TRENDS

CONCLUDE WITH GENERAL T/DAQ TRENDS

- ➡ **Increasing readout channels, and front-end cards, distributed in multi-level three structure**

CONCLUDE WITH GENERAL T/DAQ TRENDS

- **Increasing readout channels, and front-end cards, distributed in multi-level three structure**
- **Deal with dataflow instead of latency**
  - **decouple** DAQ from High Level Triggers
  - decouple dataflow from storage, with temporary buffers
  - Use COTS network and processing

CONCLUDE WITH GENERAL T/DAQ TRENDS

- ➡ **Increasing readout channels, and front-end cards, distributed in multi-level three structure**
- ➡ **Deal with dataflow instead of latency**
  - ➡ **decouple** DAQ from High Level Triggers
  - ➡ decouple dataflow from storage, with temporary buffers
  - ➡ Use COTS network and processing
- ➡ **Increase data aggregation at the Event Building**
  - ➡ reducing request rates on DAQ software
  - ➡ per-time-frame, per-orbit instead of per-event

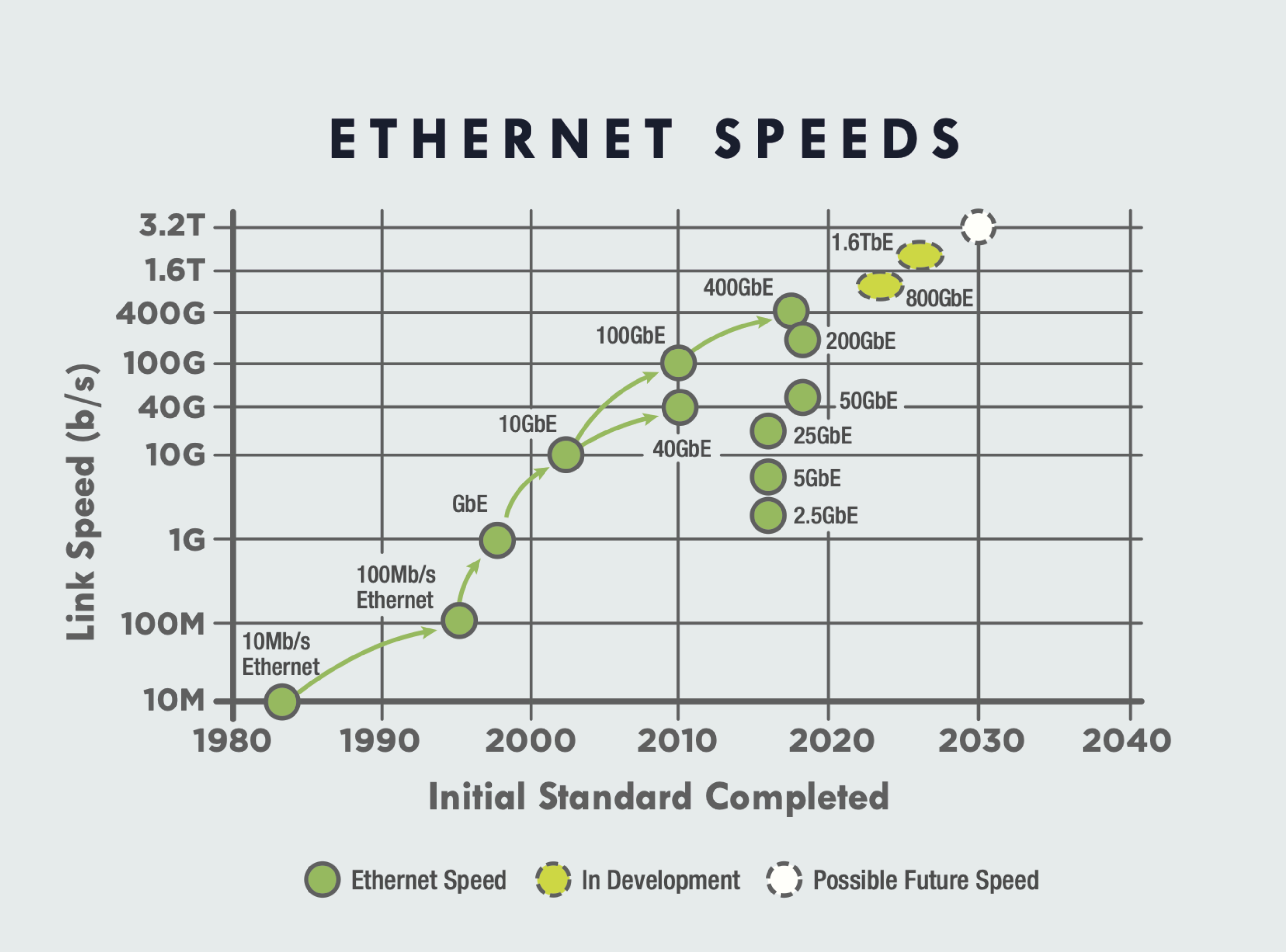
CONCLUDE WITH GENERAL T/DAQ TRENDS

- **Increasing readout channels, and front-end cards, distributed in multi-level three structure**
- **Deal with dataflow instead of latency**
  - **decouple** DAQ from High Level Triggers
  - decouple dataflow from storage, with temporary buffers
  - Use COTS network and processing
- **Increase data aggregation at the Event Building**
  - reducing request rates on DAQ software
  - per-time-frame, per-orbit instead of per-event
- **Use networks as soon as possible**
  - toward commercial bidirectional point-to-multipoint architecture

CONCLUDE WITH GENERAL T/DAQ TRENDS

- ➡ **Increasing readout channels, and front-end cards, distributed in multi-level three structure**
- ➡ **Deal with dataflow instead of latency**
  - ➡ **decouple** DAQ from High Level Triggers
  - ➡ decouple dataflow from storage, with temporary buffers
  - ➡ Use COTS network and processing
- ➡ **Increase data aggregation at the Event Building**
  - ➡ reducing request rates on DAQ software
  - ➡ per-time-frame, per-orbit instead of per-event
- ➡ **Use networks as soon as possible**
  - ➡ toward commercial bidirectional point-to-multipoint architecture
- ➡ **Use "network" design already at small scale**
  - ➡ easily get high performance with commercial components

CLEAR WHY?



NEXT ETHERNET ERA
ethernet alliance

ISOTDAQ 2024

14th International School of Trigger and Data Acquisition

Jun. 19th – Jun. 28th 2024
University of Science and Technology of China, Hefei, China

Target Audience: Physicists, engineers and computer scientists with an interest in trigger and data acquisition systems

Places are limited: Acceptance is by a selection committee

Registration Deadline: 31 January 2024

Email:
isotdaq.schools@cern.ch

Website:
https://indico.cern.ch/event/1337180

☑Trigger

- Modular Electronics
- Front-End Electronics
- Associative memories

☑DAQ

- ADC, TDC, Detector Readout
- Micro Controllers
- VMEBus, xCTA, PCI, PCIe

☑Application Examples

- General Concepts for TDAQ
- Insight on LHC TDAQ
- Non-LHC Systems

Readout
Buffer
Busy
Storage
Flipflop
Trigger
HLT
Queue
DAQ
Latency
Decoding
Eventbuilding
Rate
Dataflow
Network
Bus
Derandomization
Encoding
Microcontroller
GPU
Backpressure
Datacollection
Event
Deadtime
FPGA
FIFO
Digitalization