Introduction to Data Acquisition

ISOTDAQ2013, February 1st

W.Vandelli CERN/PH-ATD

Outline

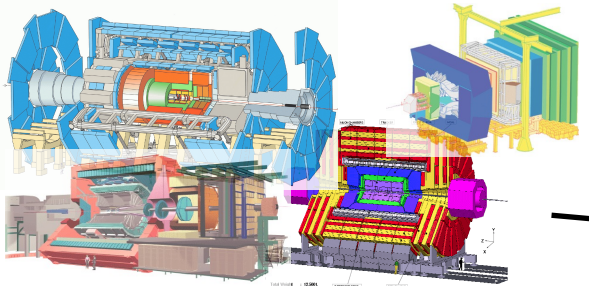
- Introduction
  - What DAQ is?
  - Overall framework
- Basic DAQ concepts
  - Digitization, Latency
  - Deadtime, De-randomization
- Scaling up
  - Readout & Event Building
  - Buses vs Networks
- Do it Yourself

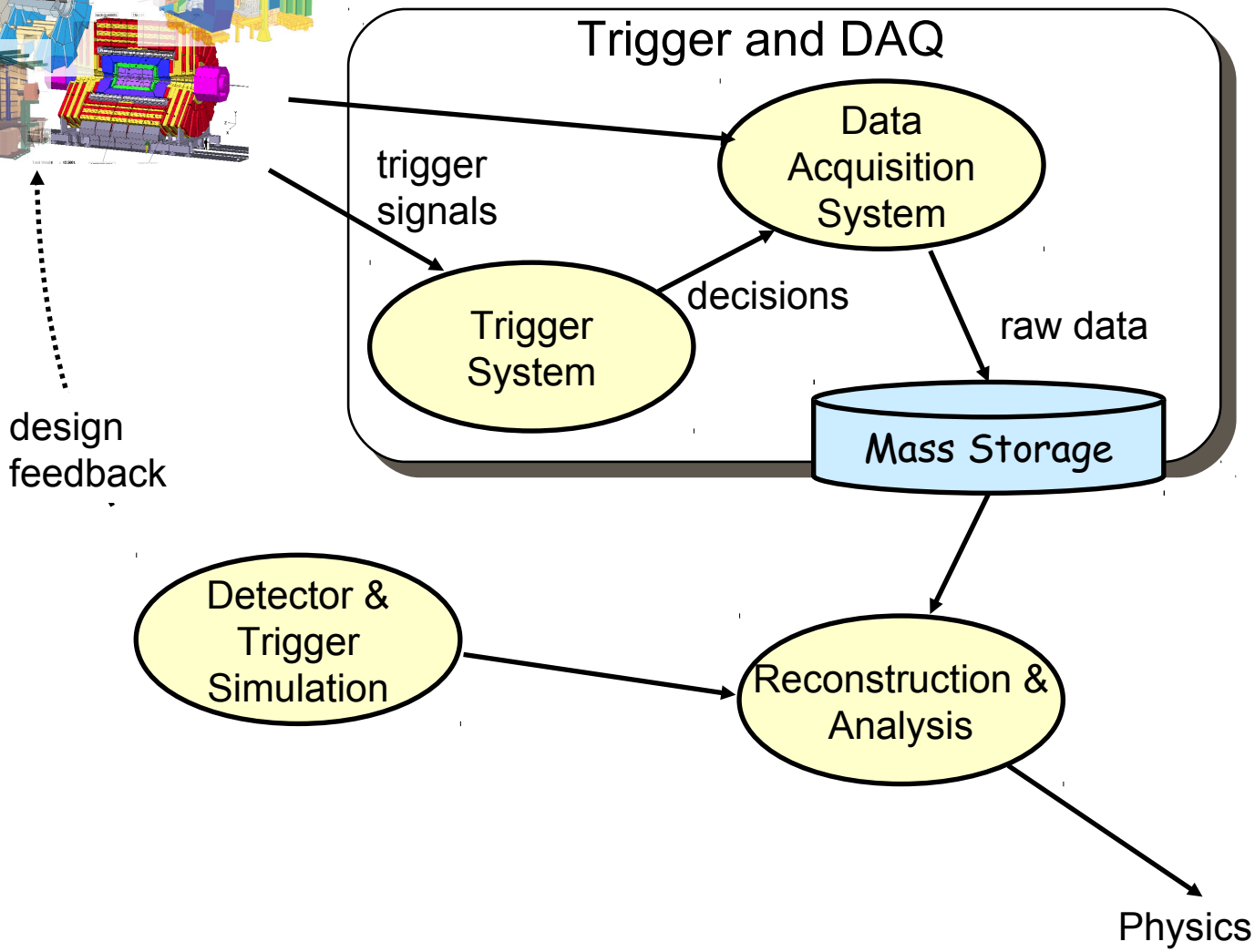
Introduction

- Data acquisition is **not an exact science**. It is an alchemy of electronics, computer science, networking, physics and … hacking and experience
  - …, money and manpower matter as well, …
- I will mostly refer to DAQ in High-Energy Physics
- Aim of this lesson is to introduce the basic DAQ concepts avoiding as many technological details as possible. The following courses will cover these aspects

Material and ideas have been taken from CERN Summer Student lectures (N.Neufeld and C.Gaspar) and from the "Physics data acquisition and analysis" lessons given by R.Ferrari at the University of Parma, Italy

General Overview





➔Overall the main role of T & DAQ is to process the signals generated in a detector, storing the interesting information on a permanent storage

Trigger & DAQ

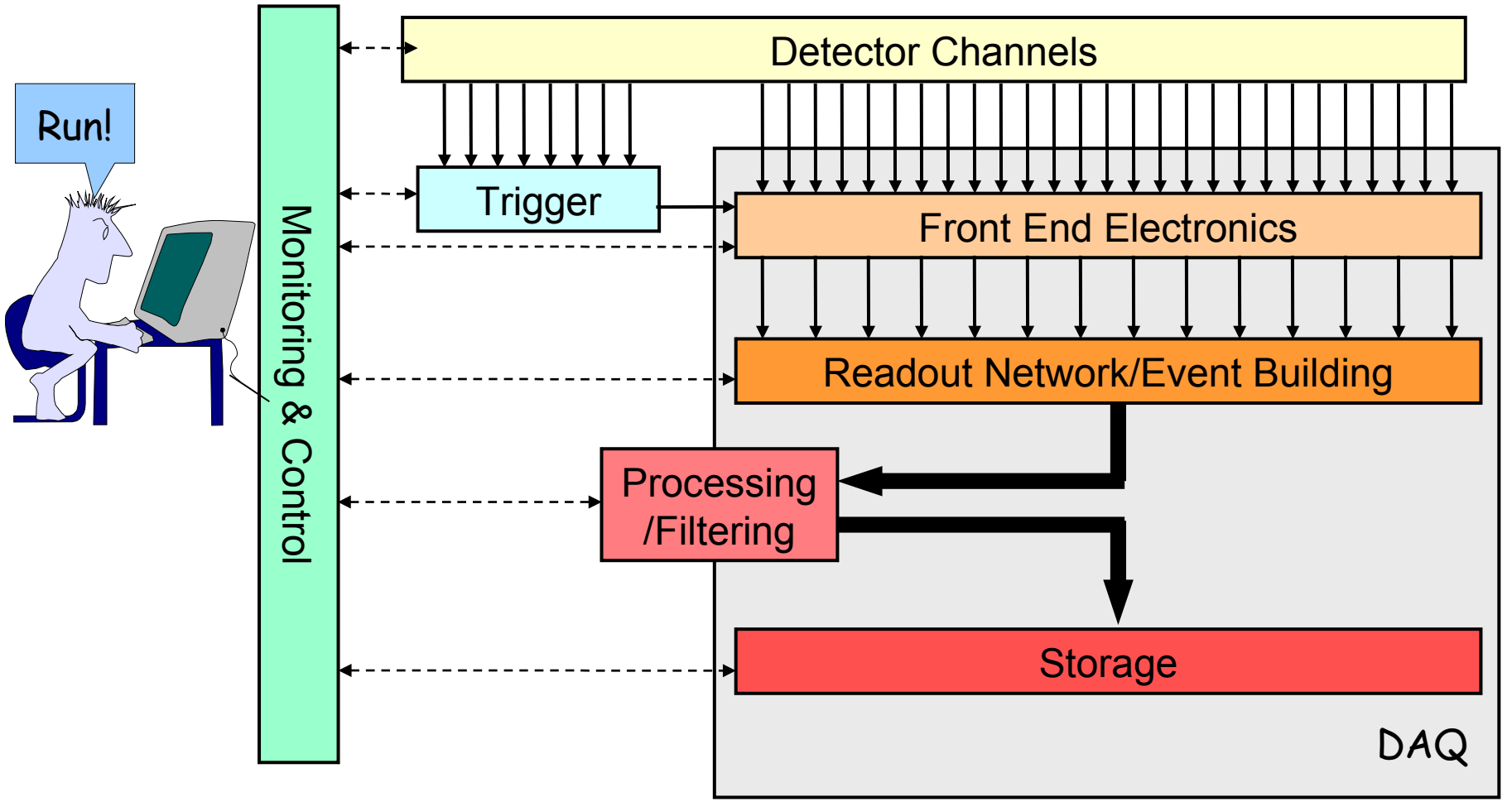
➔Trigger:

- Either selects interesting events or rejects boring ones, in real time (i.e. with minimal *controlled* latency)

➔DAQ:

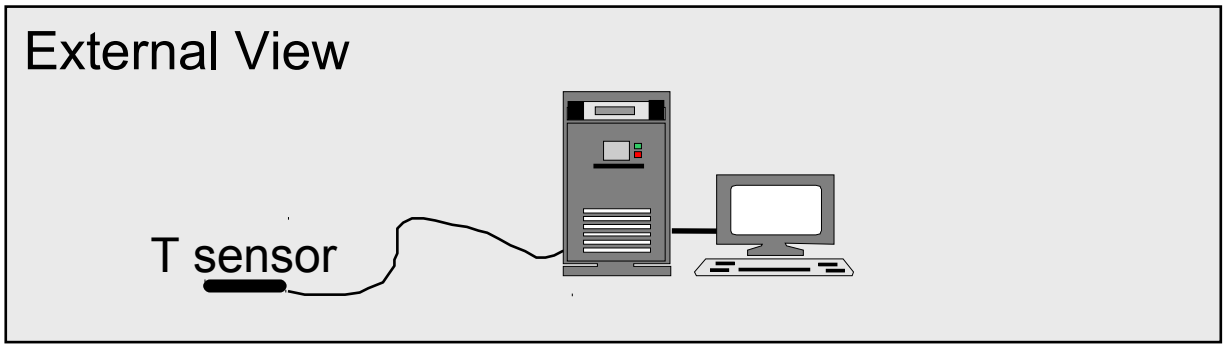
- Gathers the data produced by the detectors (Readout)
- Possibly feeds several trigger levels
- Forms complete events (Event Building)
- Stores event data (Data logging)
- Provides control, configuration and monitoring facilities (Run Control, Monitoring)

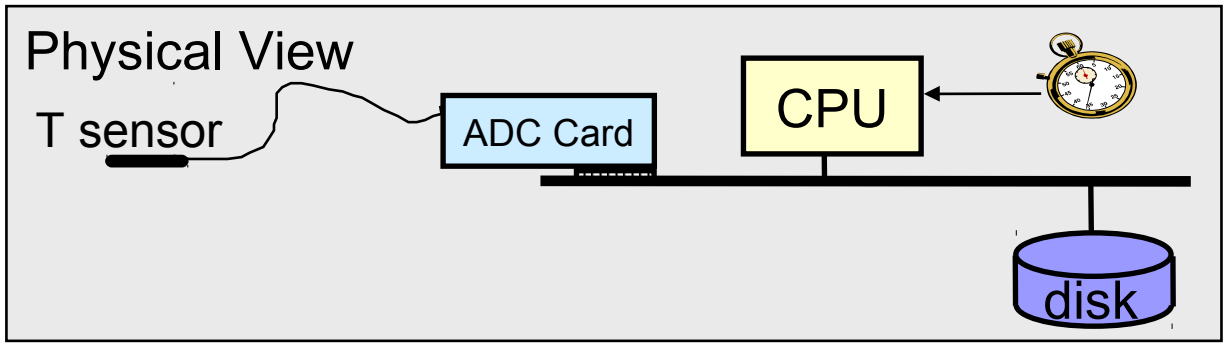
Trigger, DAQ & Controls

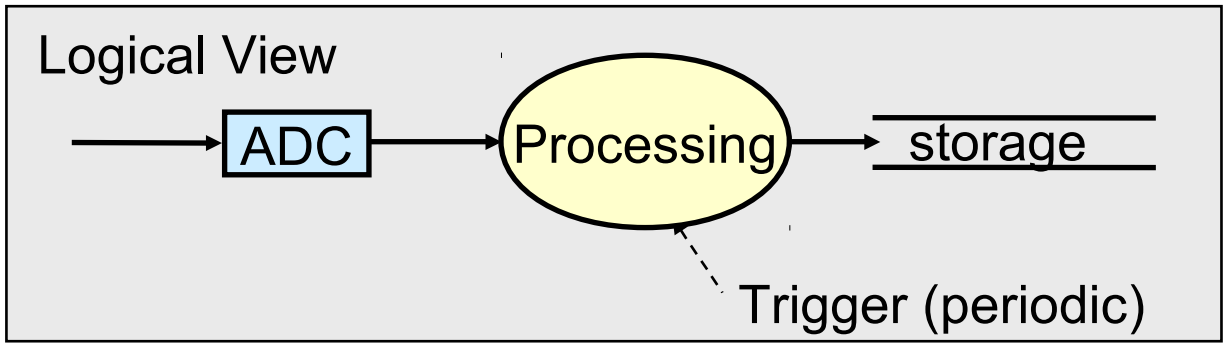


DAQ Concepts

Basic DAQ: periodic trigger

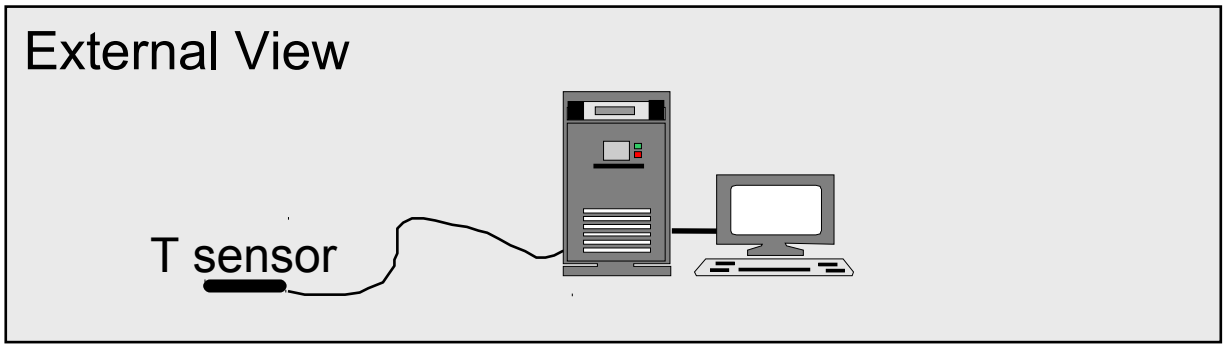


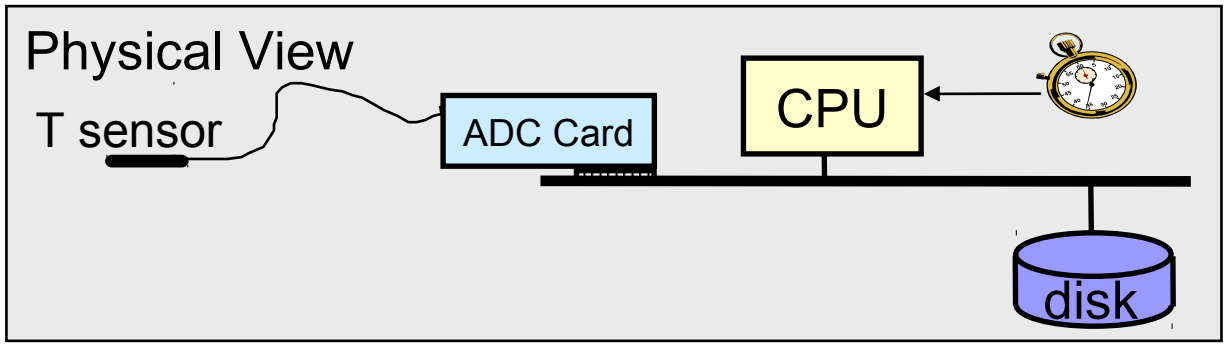


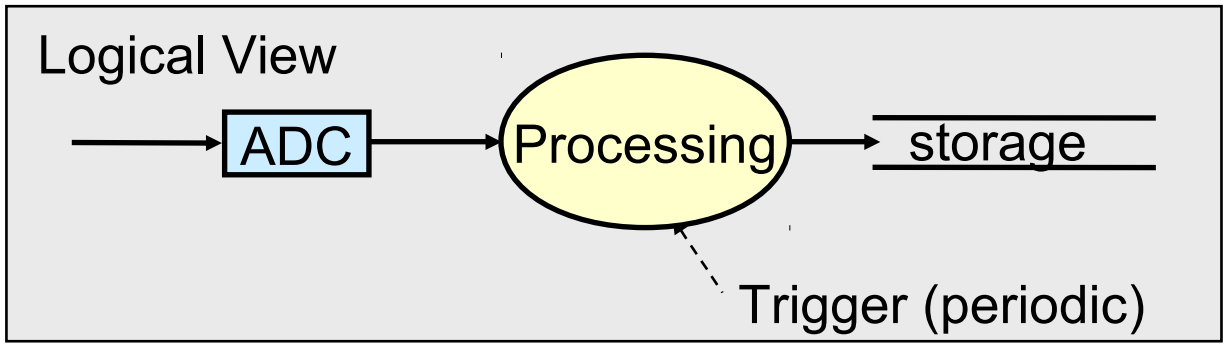


- ➔Measure temperature at a fixed frequency
- ➔ADC performs analog to digital conversion (digitization)
  - Our front-end electronics
- ➔CPU does readout and processing

Basic DAQ: periodic trigger







➔Measure temperature at a fixed frequency

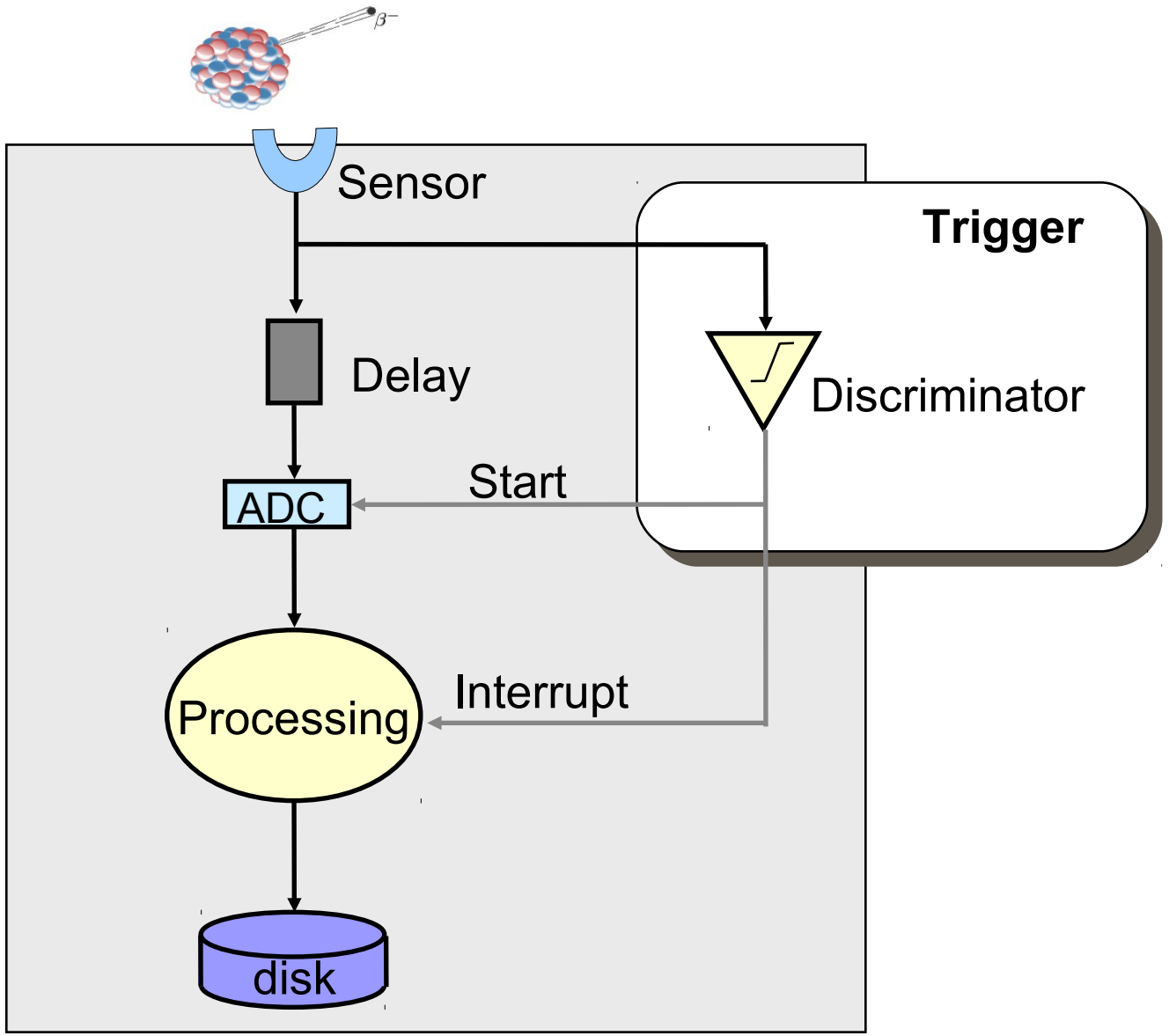
➔The system is clearly limited by the time to process an "event"

➔Example τ=1ms to

- ADC conversion
  +CPU processing
  +Storage

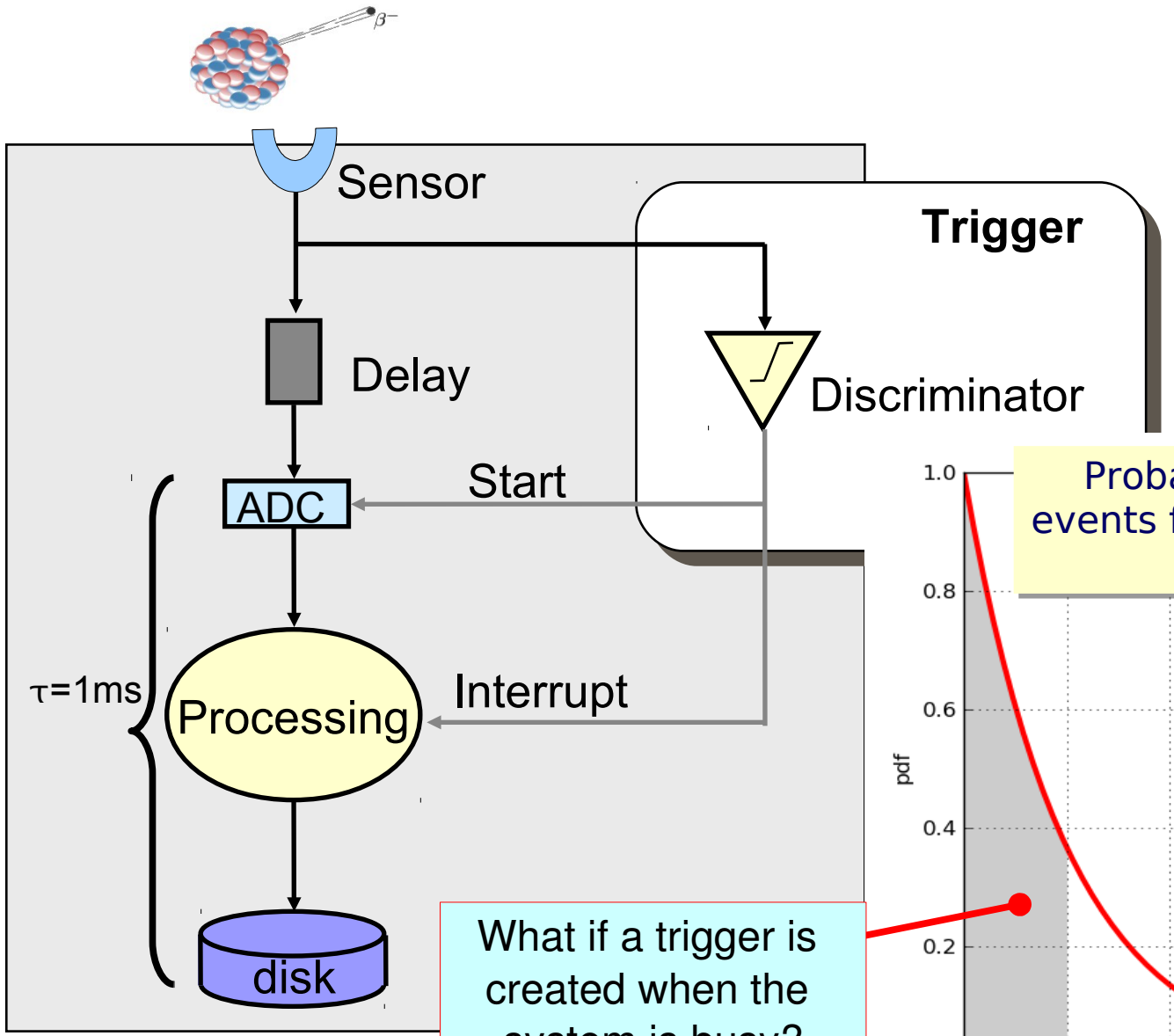
➔Sustain ~1/1ms=1kHz ***periodic trigger*** rate

Basic DAQ: real trigger



- ➔Measure β decay properties
- ➔Events are asynchronous and unpredictable
  - Need a **physics** trigger
- ➔Delay compensates for the trigger latency

Basic DAQ: real trigger



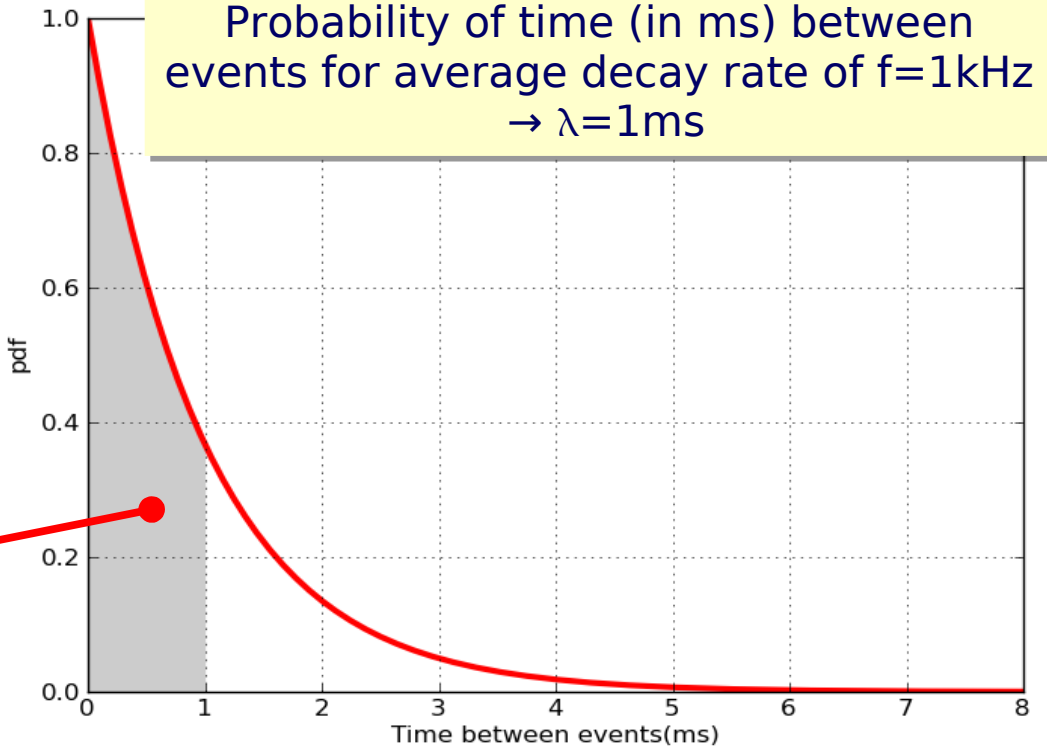
➔Measure β decay properties

- Need a **physics** trigger

➔Stochastic process

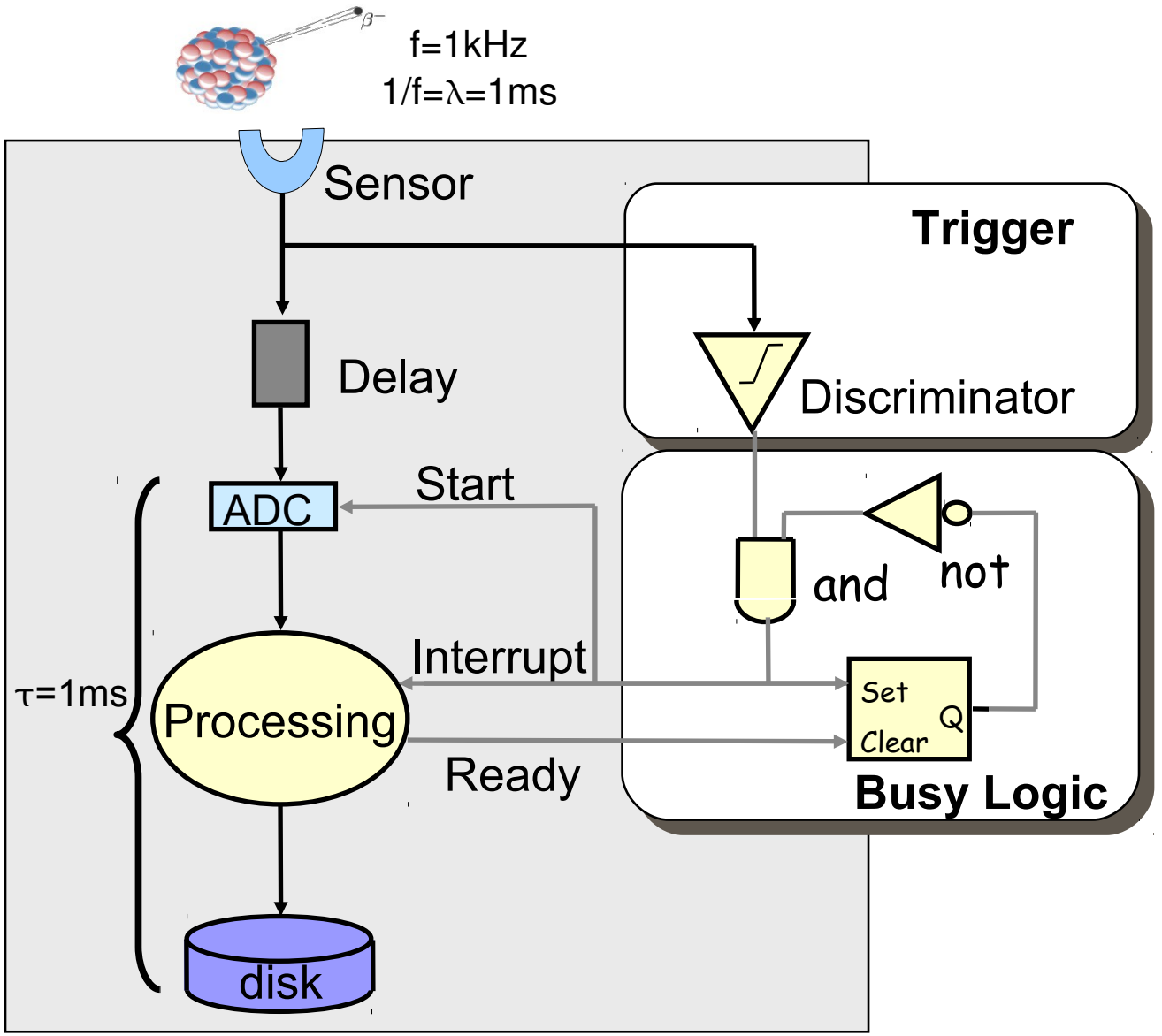
- Fluctuations

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



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

Basic DAQ: real trigger & busy logic



- ➔Busy logic avoids triggers while processing
- ➔Which (average) DAQ rate can we achieve now?
  - Reminder: τ=1ms was sufficient to run at 1kHz with a clock trigger

DAQ Deadtime & Efficiency (1)

Define ν as average DAQ frequency

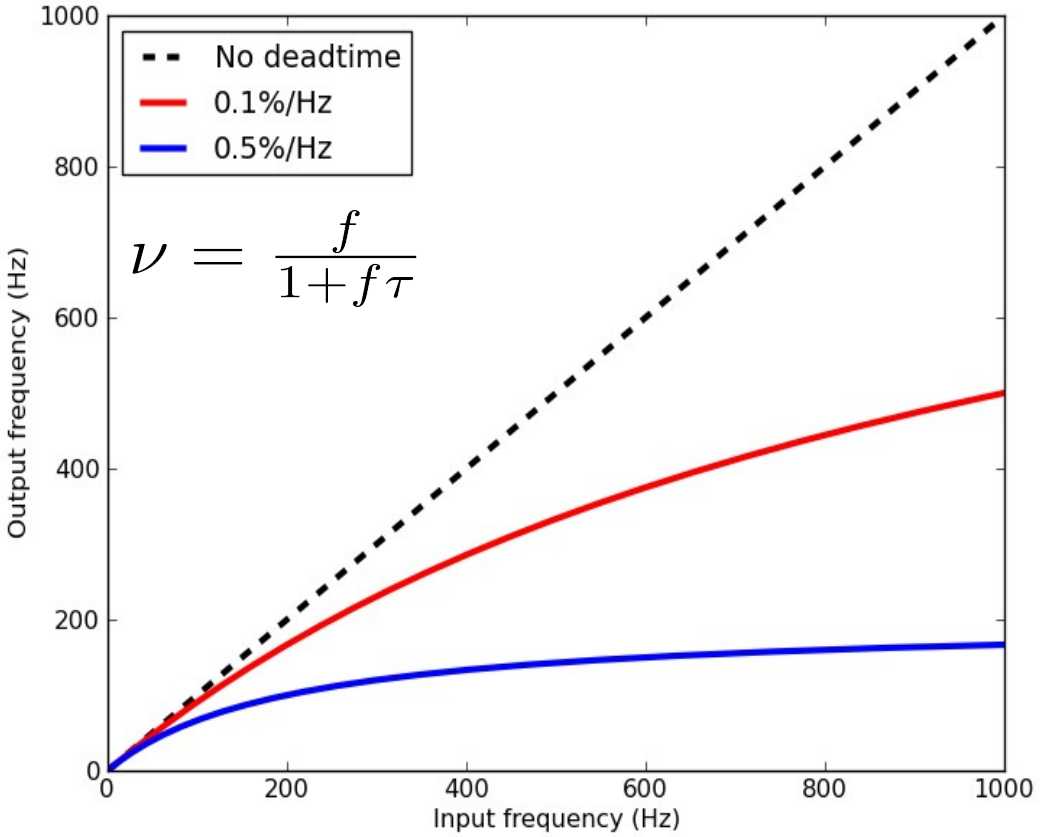
$\nu\tau \rightarrow$ DAQ system is busy - $(1-\nu\tau) \rightarrow$ DAQ system is free

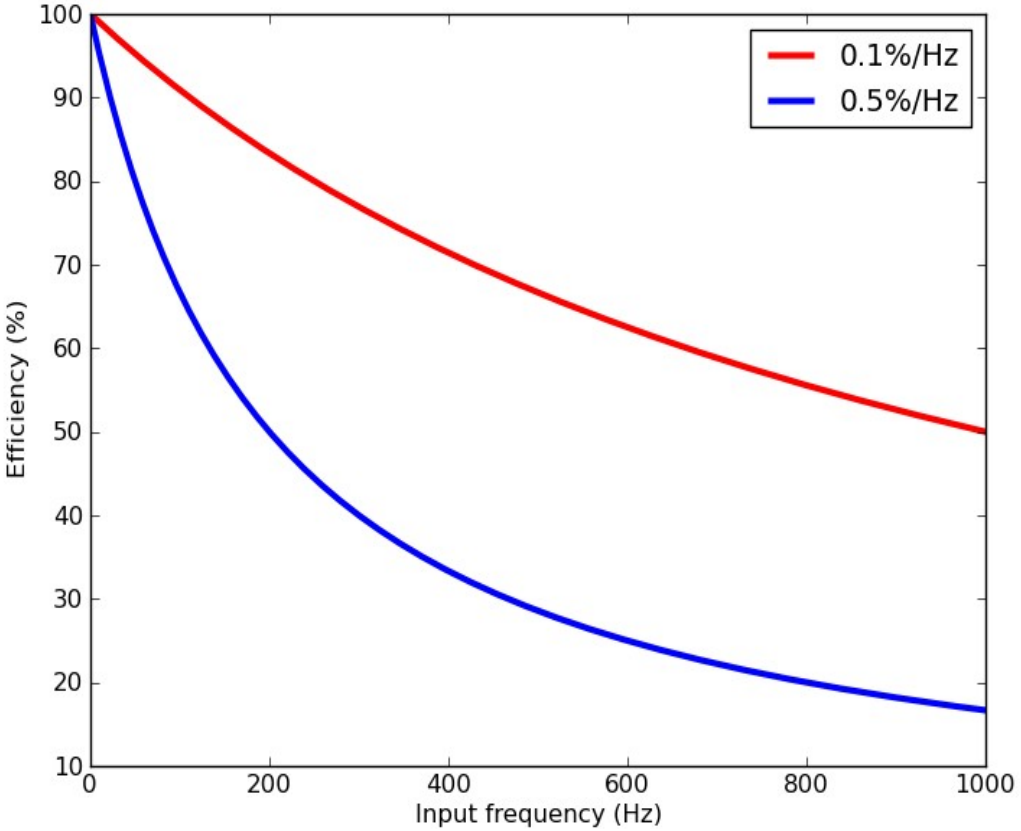
$$f(1-\nu\tau) = \nu \rightarrow \nu = \frac{f}{1+f\tau} < f$$

$$\epsilon = \frac{N_{saved}}{N_{tot}} = \frac{1}{1+f\tau} < 100\%$$

- ➔ Define DAQ deadtime (d) as the time the system requires to process an event, without being able to handle other triggers. In our example d=0.1%/Hz
- ➔ Due to the fluctuations introduced by the stochastic process the efficiency will always be less 100%
  - In our specific example, d=0.1%/Hz, f=1kHz → ν=500Hz, ϵ=50%

DAQ Deadtime & Efficiency (2)



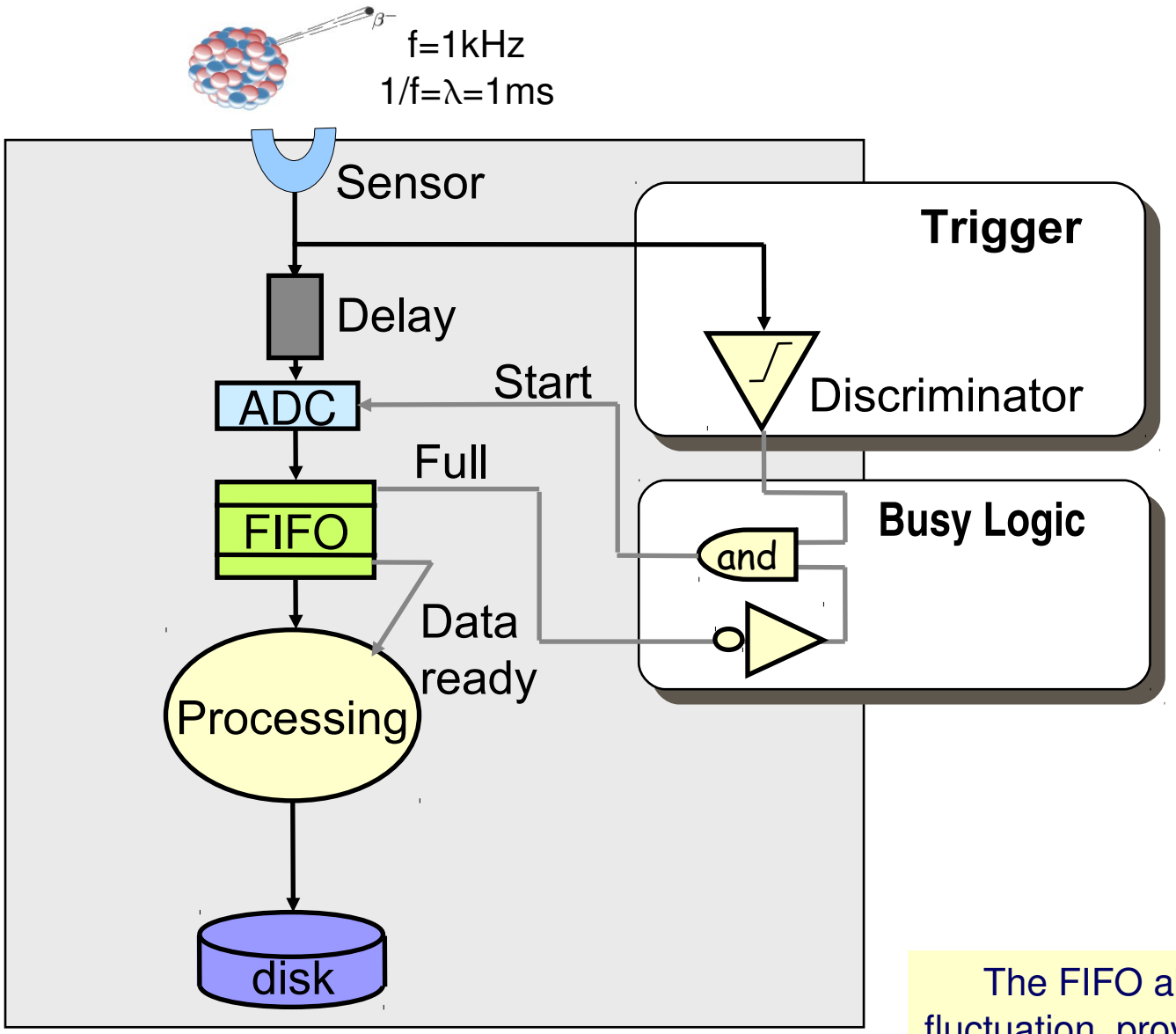


➔ If we want to obtain $\nu \sim f$ ($\epsilon \sim 100\%$) → $f\tau \ll 1$ → $\tau \ll \lambda$

- f=1kHz, ϵ=99% → τ<0.1ms → $1/\tau$>10kHz

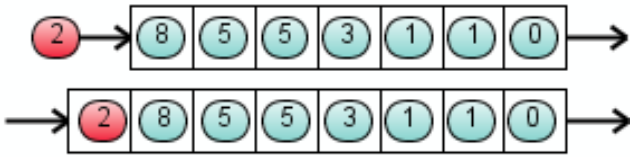
➔ In order to cope with the input signal fluctuations, we have to over-design our DAQ system by a factor 10. This is very inconvenient! Can we mitigate this effect?

Basic DAQ: De-randomization



➔First-In First-Out

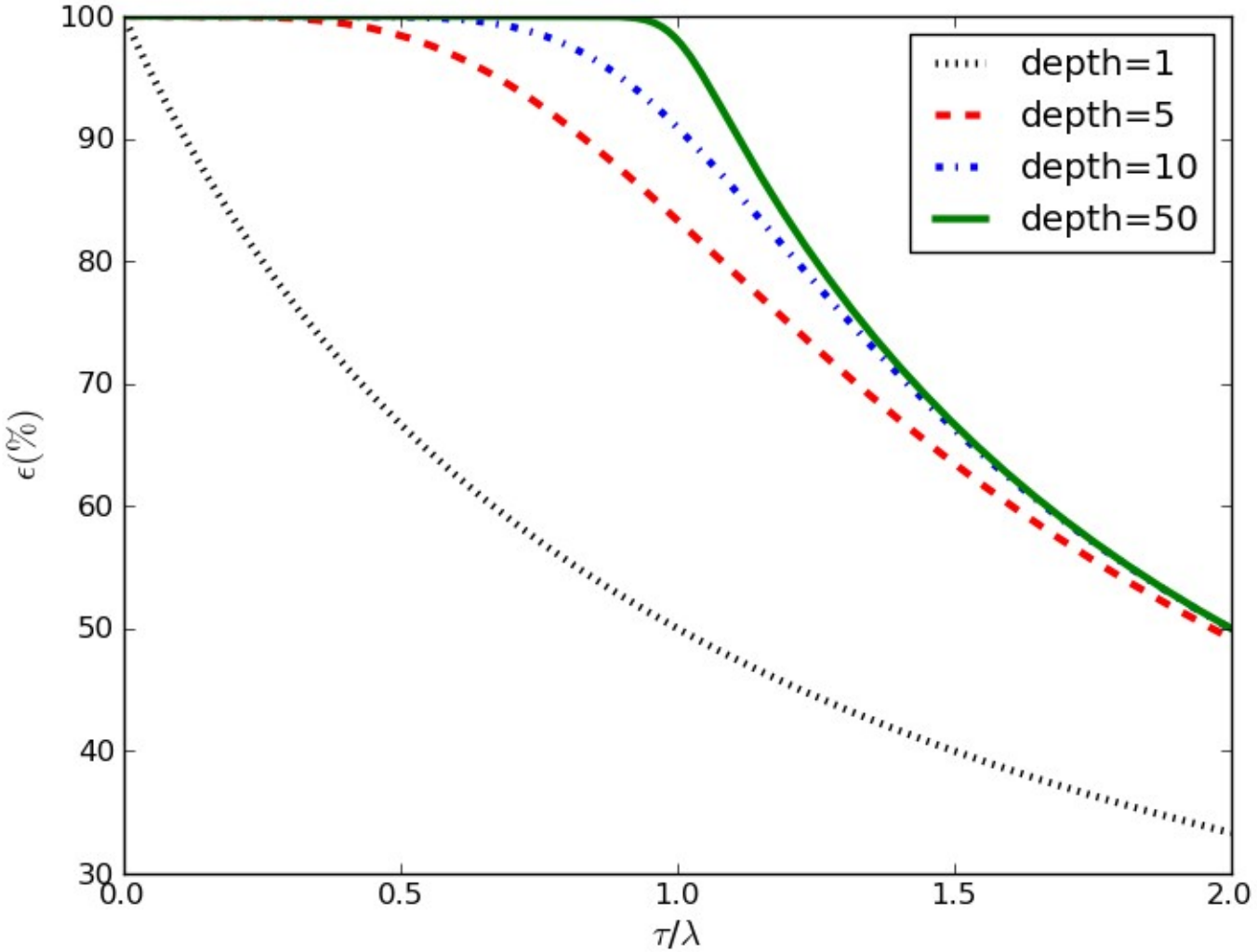
- Buffer area organized as a queue
- Depth: number of cells
- Implemented in HW and SW

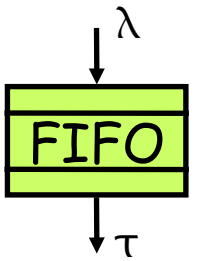


➔FIFO introduces an additional latency on the data path

The FIFO absorbs and smooths the input fluctuation, providing a ~steady (De-randomized) output rate

De-randomization: queuing theory



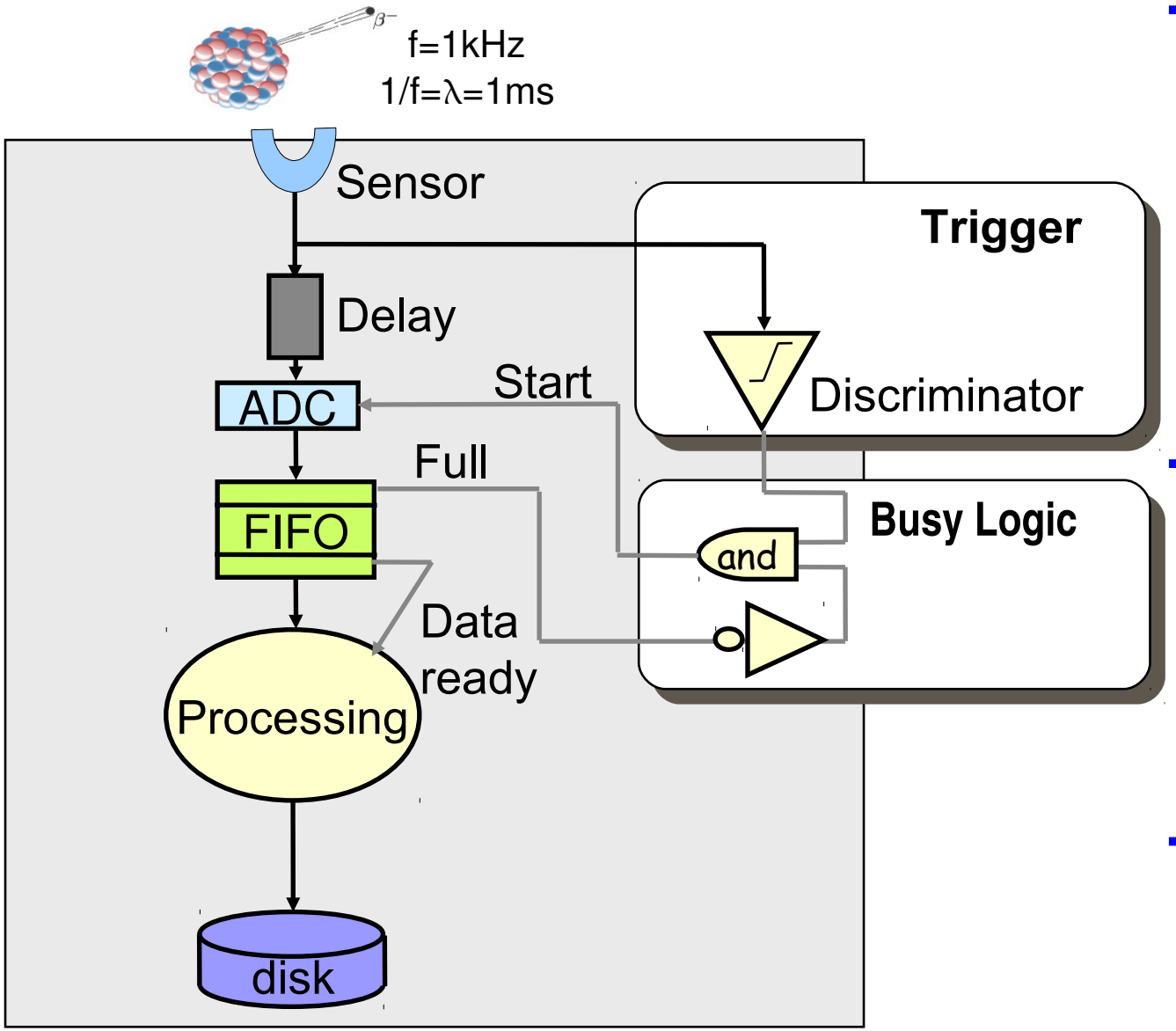


➔We can now attain a FIFO efficiency ~100% with $\tau \sim \lambda$

- Moderate buffer size

Analytic calculation possible for very simple systems only. Otherwise simulations must be used.

De-randomization: summary



➔ Almost 100% efficiency and minimal deadtime are achieved if

- ADC is able to operate at rate >>f
- Data processing and storing operates at ~f

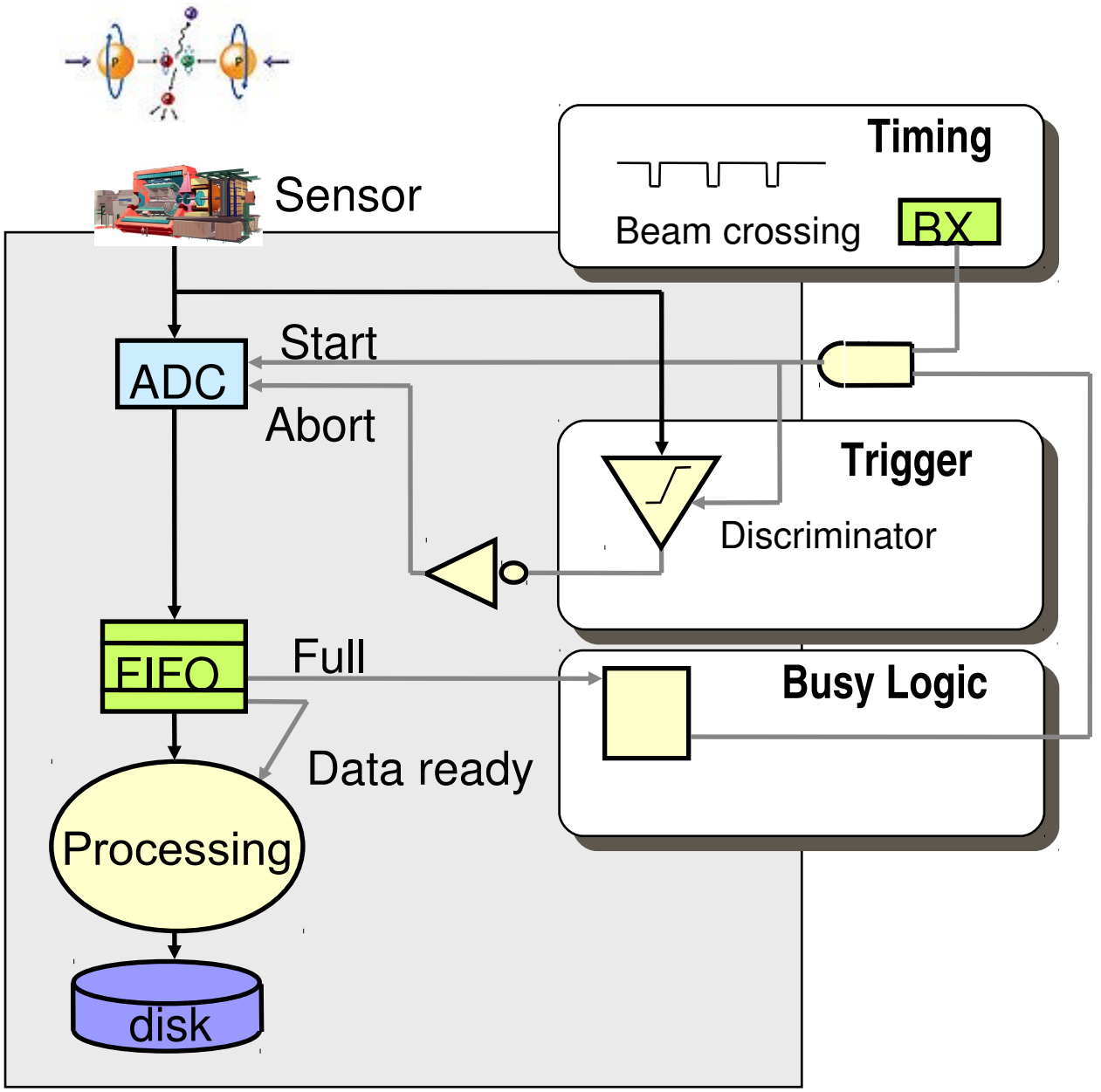
➔ The FIFO decouples the low latency front-end from the data processing

- Minimize the amount of "unnecessary" fast components

➔ Could the delay be replaced with a "FIFO"?

- Analog pipelines → Heavily used in LHC DAQs

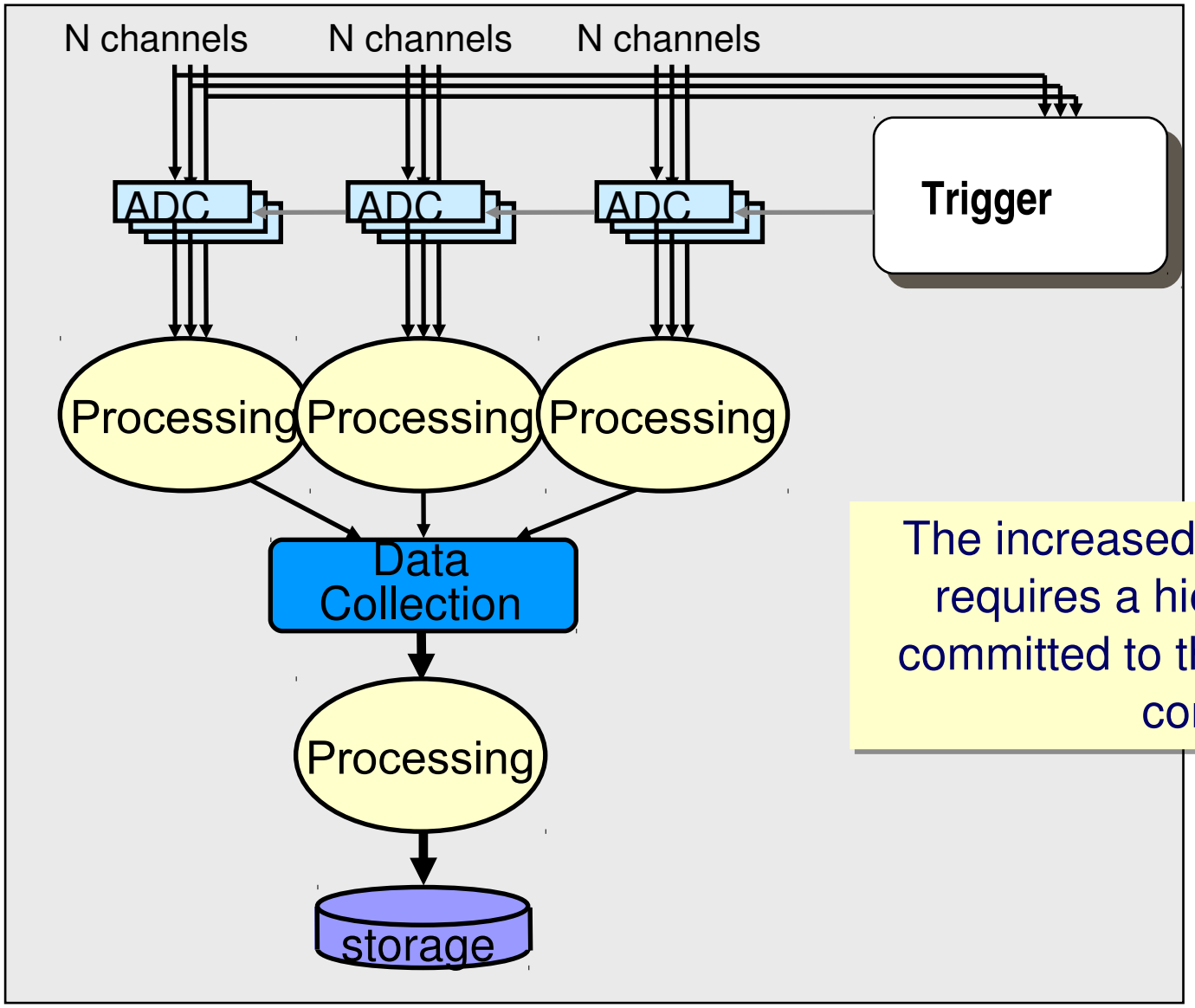
Basic DAQ: collider mode



- ➔Particle collisions are synchronous
- ➔Trigger rejects uninteresting events
- ➔Even if collisions are synchronous, the triggers (i.e. good events) are unpredictable
- ➔De-randomization is still needed

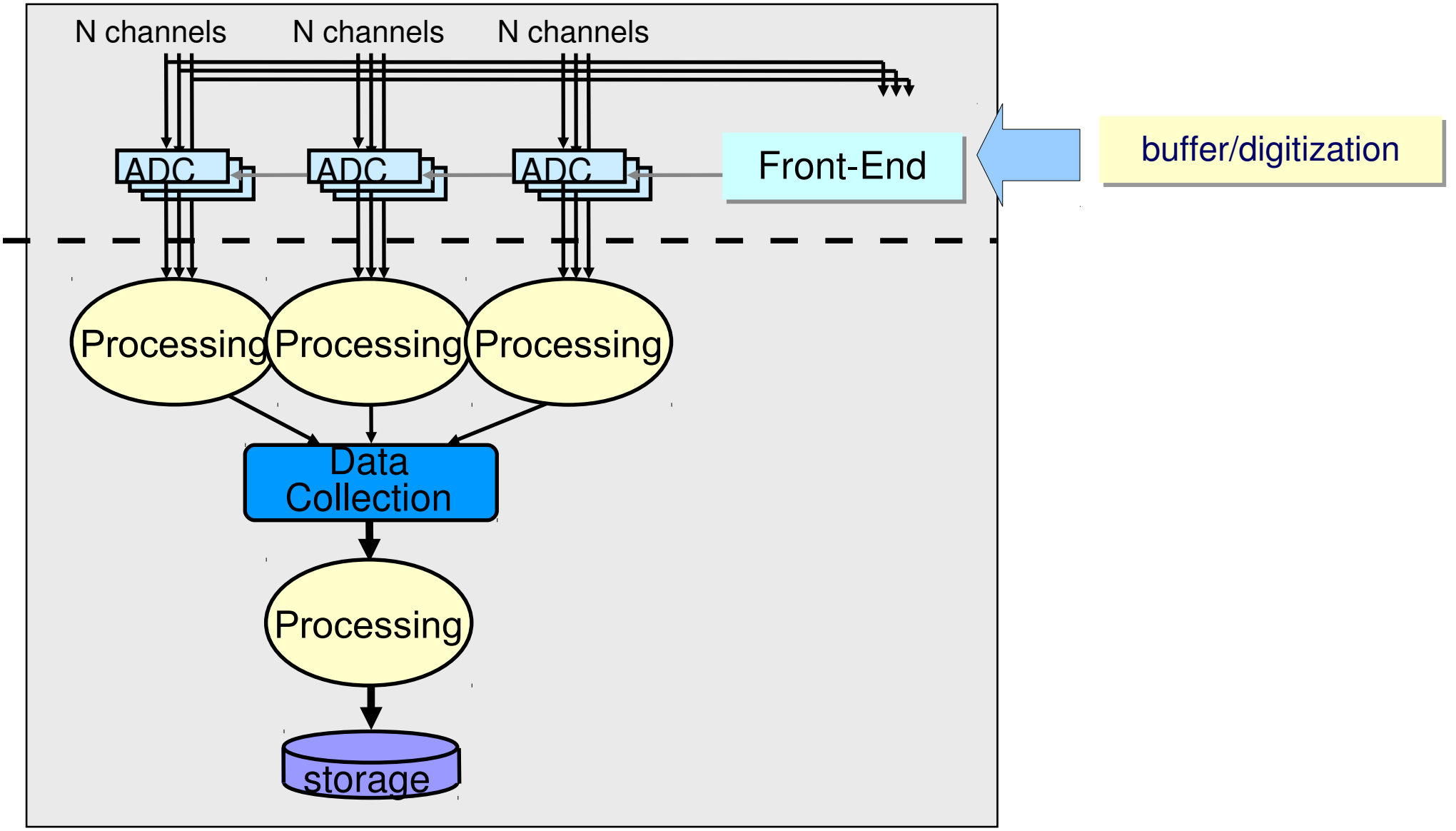
Scaling up

Basic DAQ: more channels

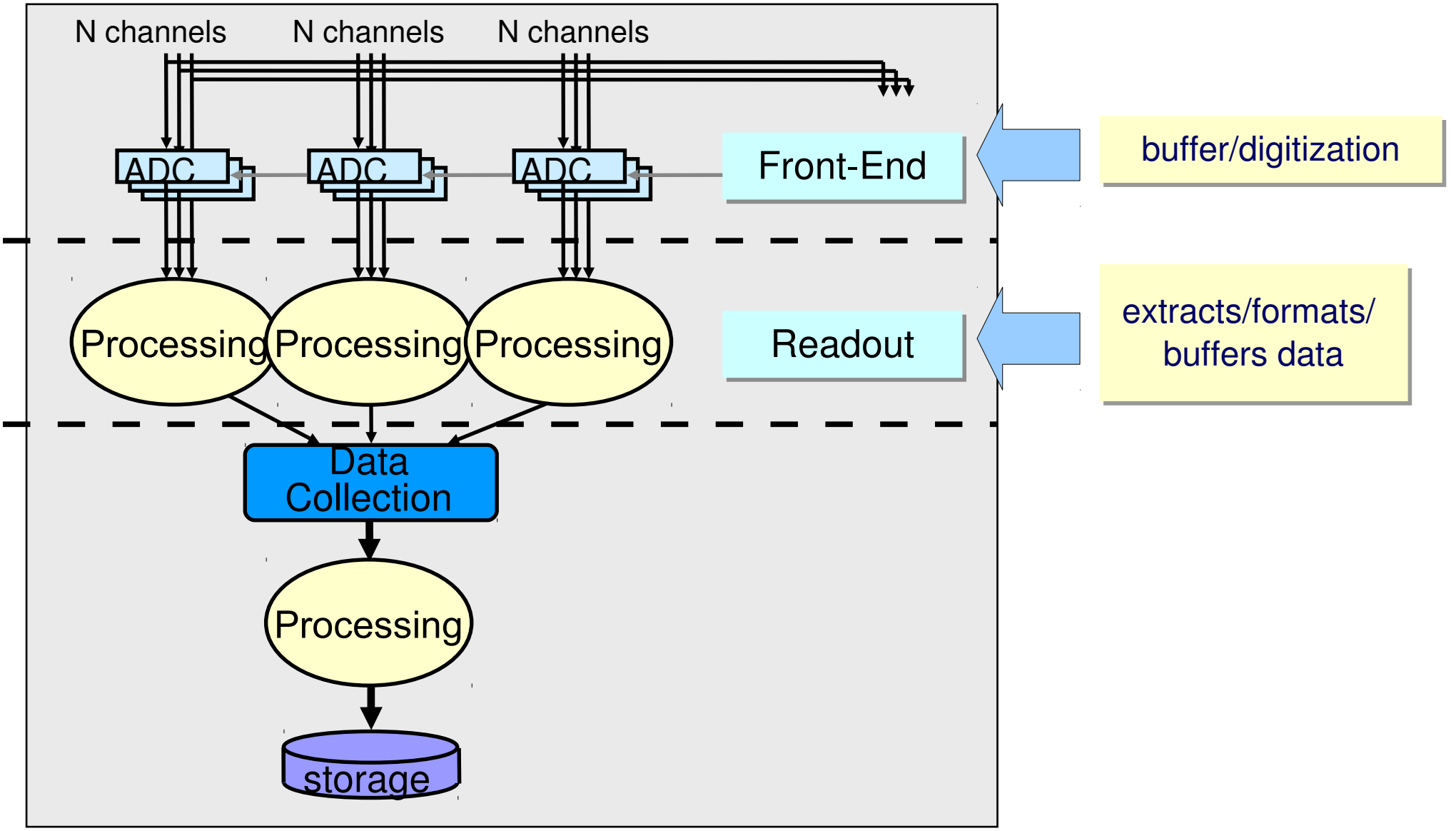


The increased number of channels requires a hierarchical structure committed to the data handling and conveyance

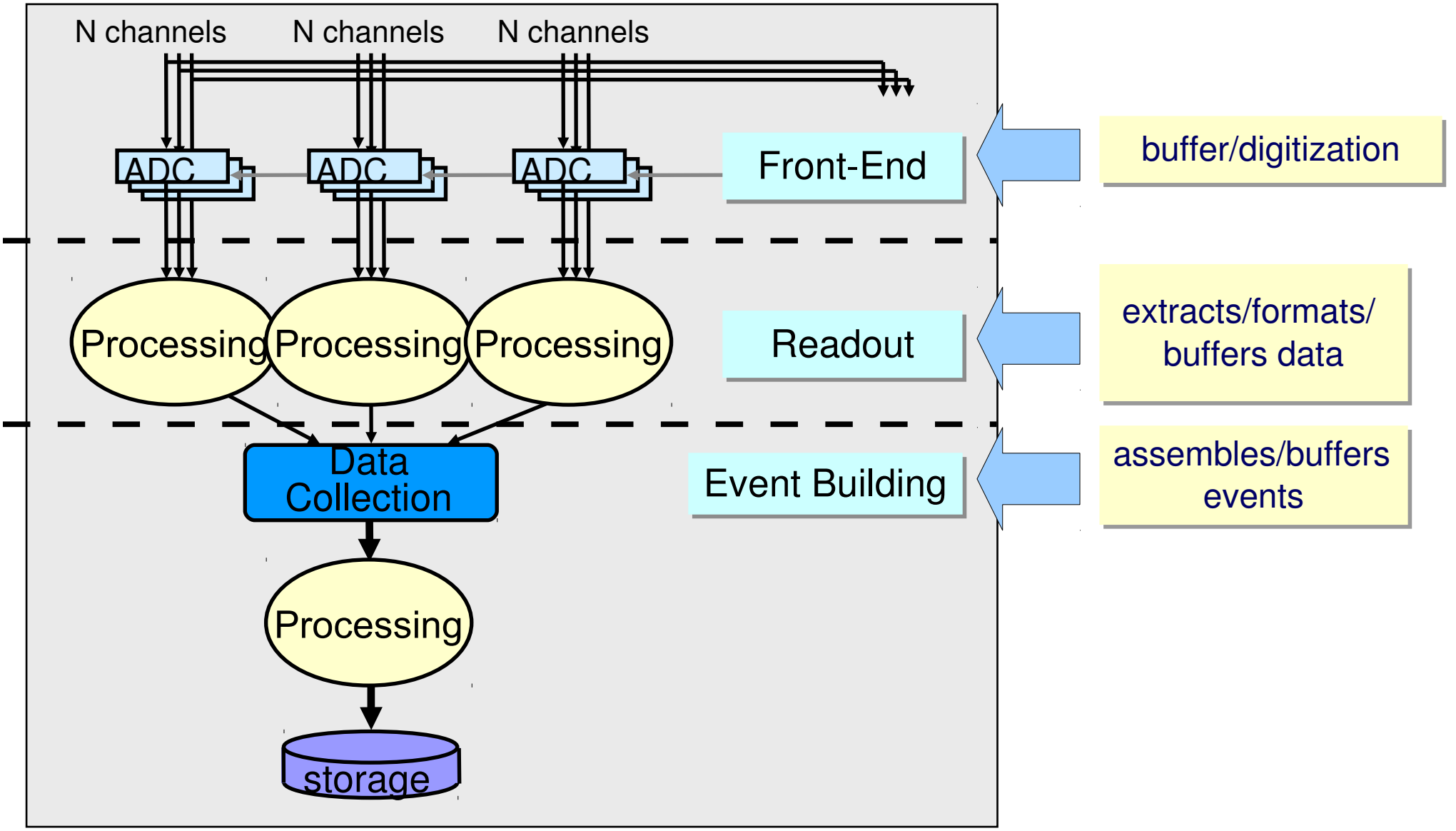
Large DAQ: Constituents



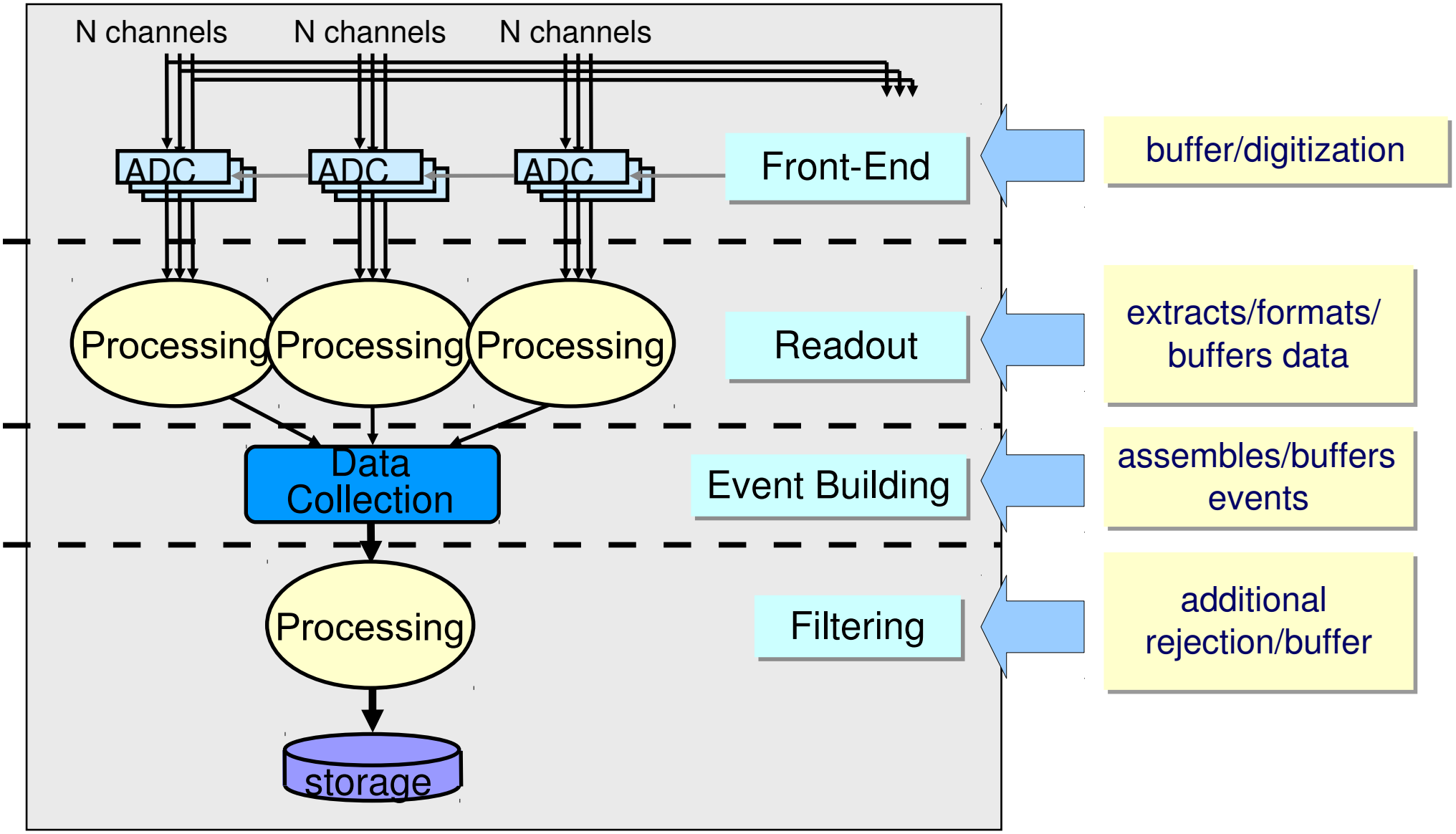
Large DAQ: Constituents



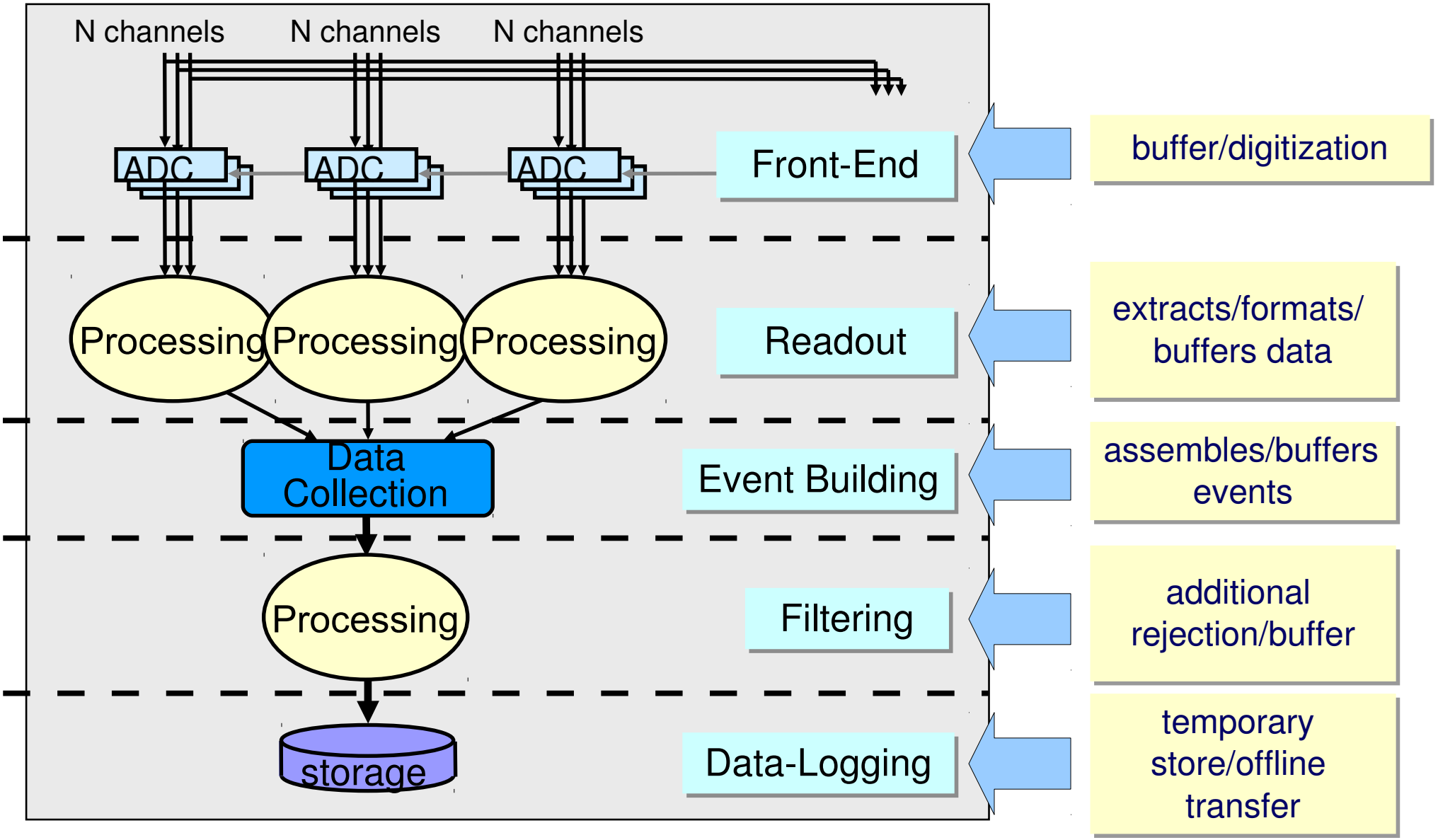
Large DAQ: Constituents



Large DAQ: Constituents

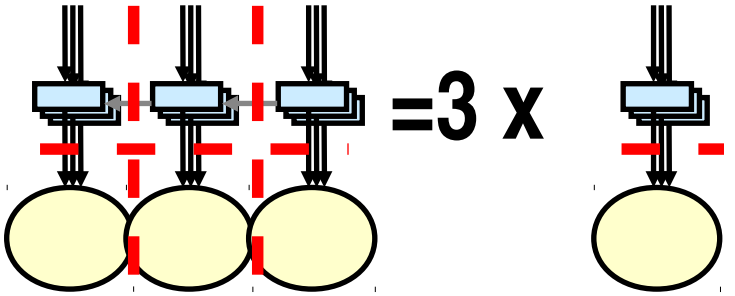


Large DAQ: Constituents

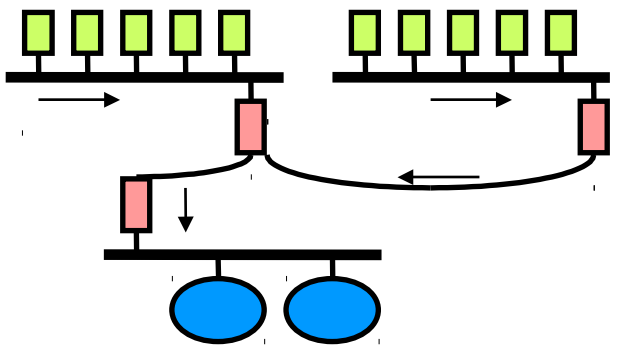


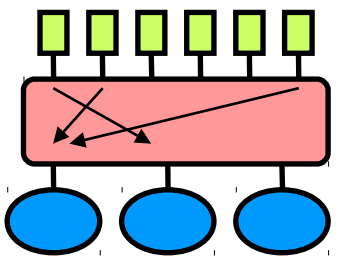
Readout Topology

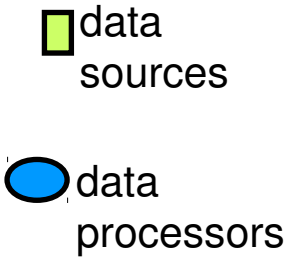
- Reading out or building events out of many channels requires many components
- In the design of our hierarchical data-collection system, we have better define "building blocks"
  - Example: readout crates, event building nodes, …



- How to organize the interconnections inside the building blocks and between building blocks?
- Two main classes: bus or network

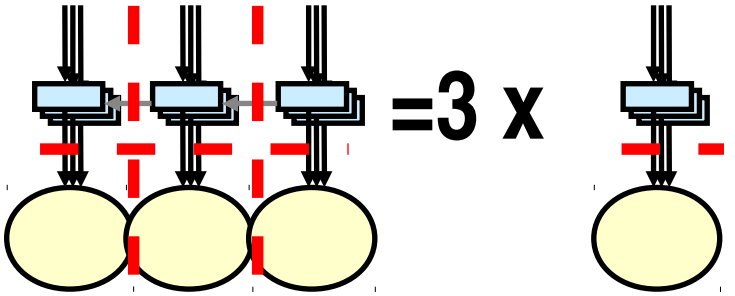






Readout Topology

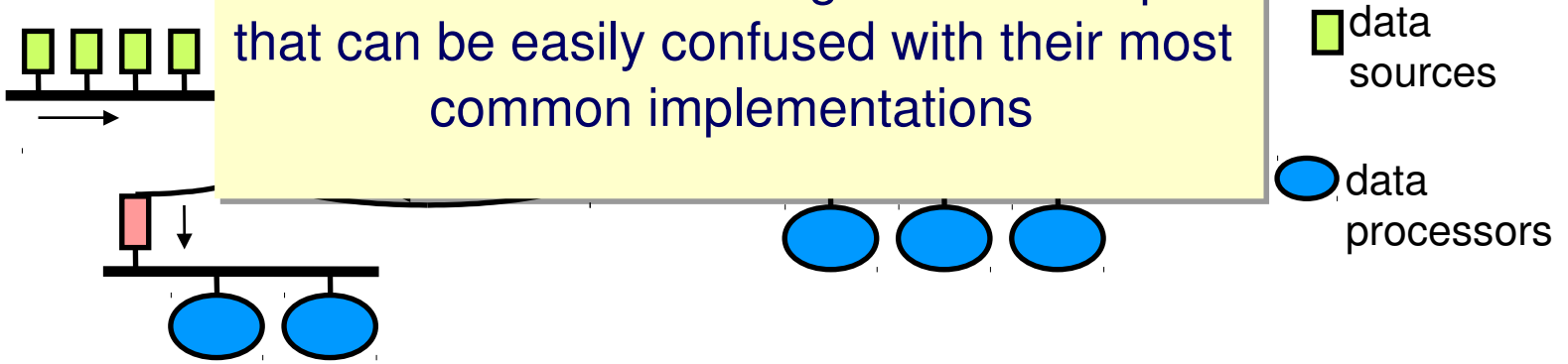
- Reading out or building events out of many channels requires many components
- In the design of our hierarchical data-collection system, we have better define "building blocks"
  - Example: readout crates, event building nodes, …



- How to organize the interconnections inside the building blocks and between building blocks?
- Two main cla

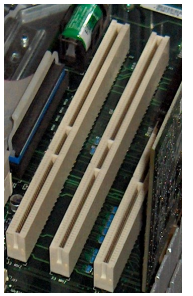
Warning

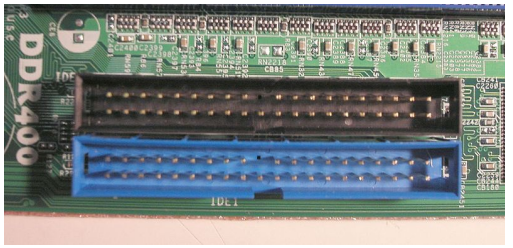
Buses and network are generic concepts that can be easily confused with their most common implementations

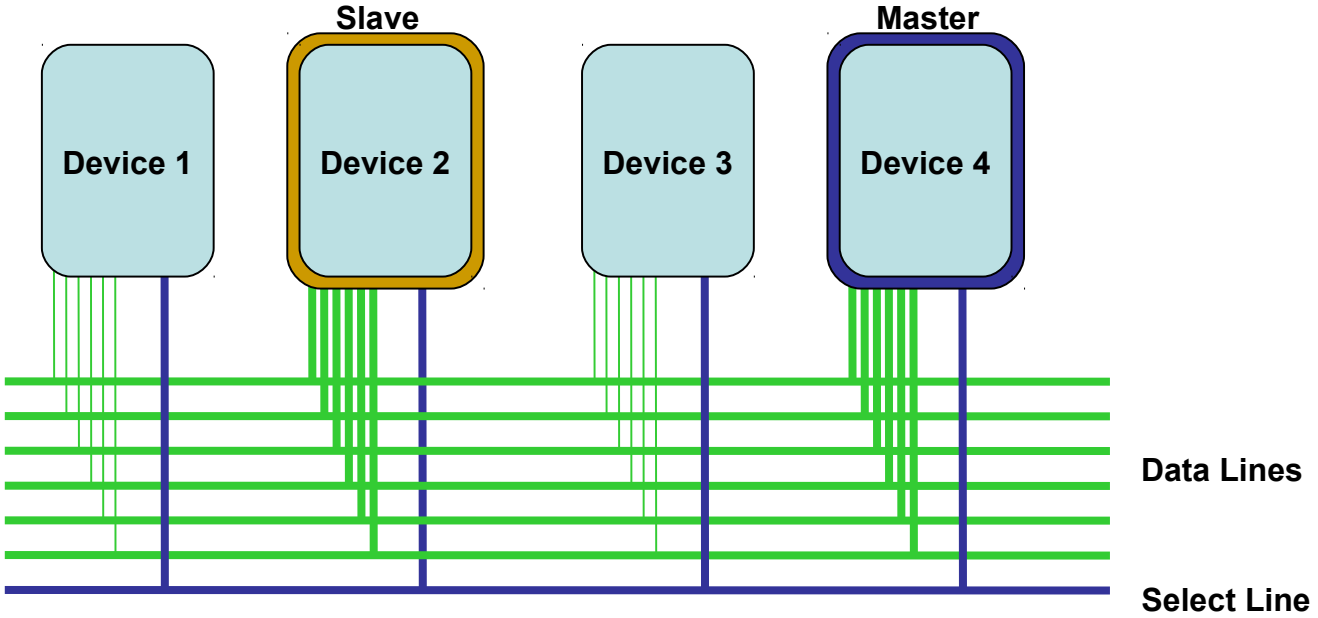


Buses

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







Bus facts

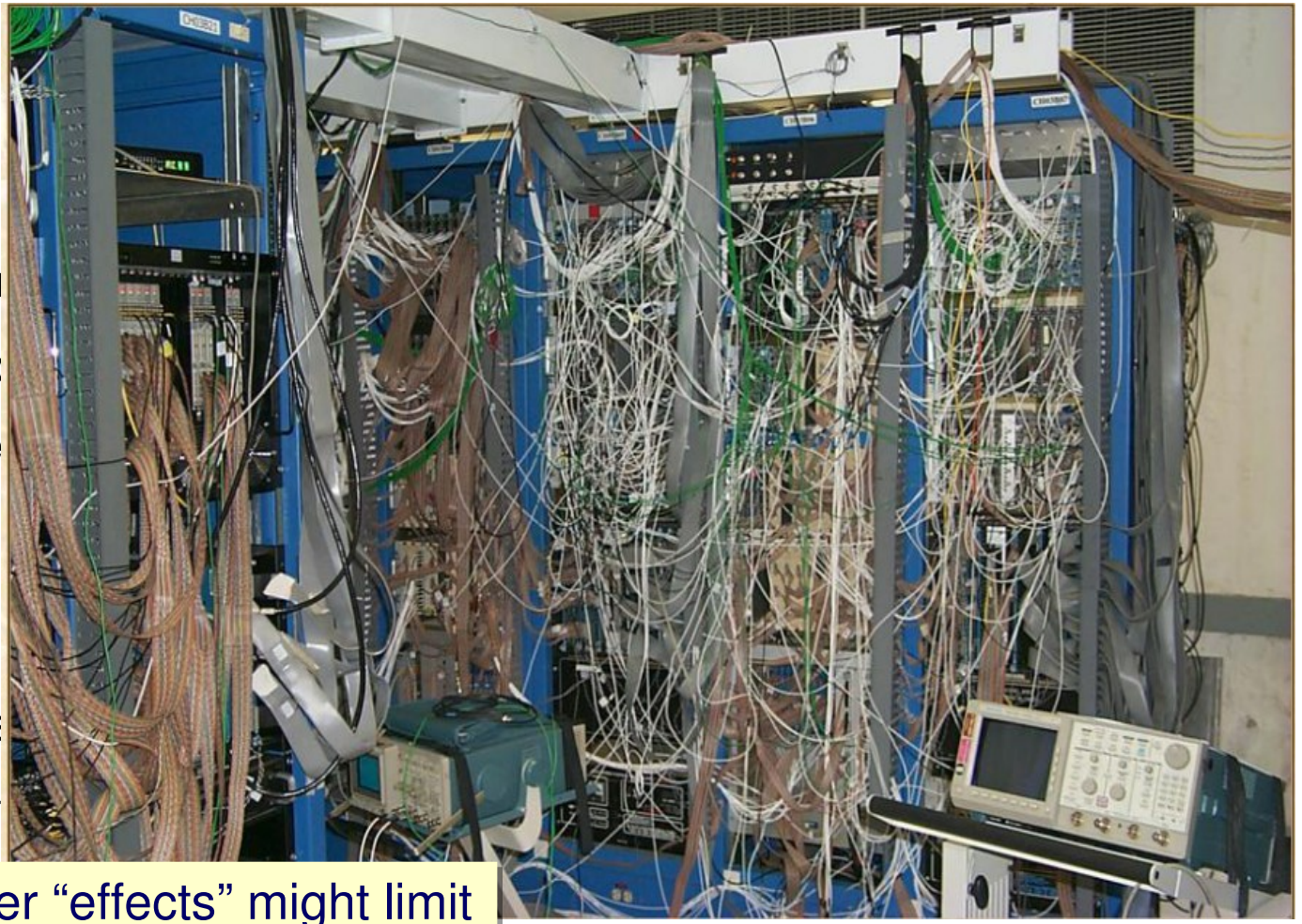
➔Simple ✔

- 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 bus frequency is inversely proportional to the bus length
- Maximum number of devices depends on the bus length

Bus facts



➔Simple ✔

- Fixed number of li
- Devices have to fo
  - Mechanical, ele

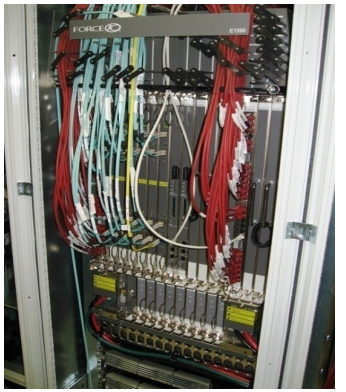
➔Scalability issues ✘

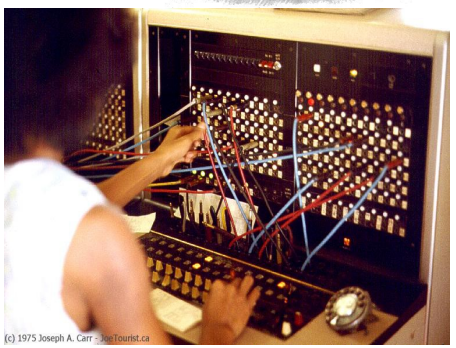
- Bus bandwidth is s
- Maximum bus wid

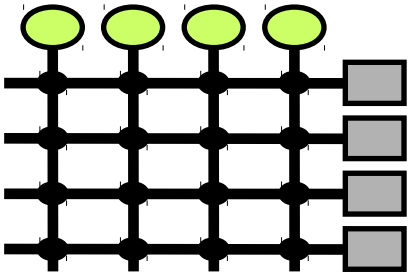
ne bus length

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

Network

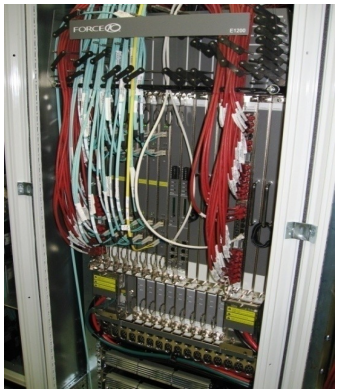


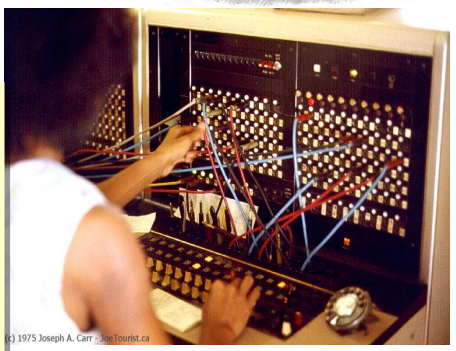


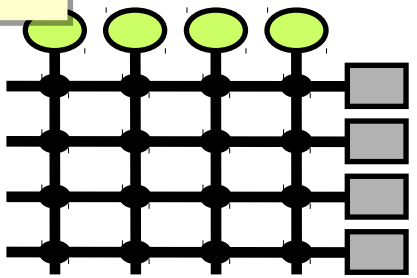


- ➔Examples: Ethernet, Telephone, Infiniband, …
- ➔All devices are equal
- ➔Devices communicate directly with each other
  - No arbitration, simultaneous communications
- ➔Device communicate by sending messages
- ➔In switched network, switches move messages between sources and destinations
  - Find the right path
  - Handle “congestion” (two messages with the same destination at the same time)
    - Would you be surprised to learn that buffering is the key?

Network







- ➔Examples: Ethernet, Telephone, Infiniband, …
- ➔All devices are equal
- ➔Devices communicate directly with each other
  - No arbitration, simultaneous communications
- ➔Device communicate by sending messages
- ➔In switched n
  sources and
  - Find the r
  - Handle “congestion” (two messages with the same destination at the same time)
    - Would you be surprised to learn that buffering is the key?

Thanks to these characteristics, networks do scale well. They are the backbones of LHC DAQ systems

Do It Yourself

DAQ Mentoring

➔Study the trigger properties

- Periodic or stochastic - Continuous or bunched - Rejecting or selecting

➔Consider the needed efficiency

- It is good to keep operation margins, but avoid over-sizing

➔Identify the fluctuation sources and size adequate buffering mechanisms

- Watch out: (deterministic) complex systems introduce fluctuations: multi-threaded software, network communications, …

➔An adequate buffer is not a huge buffer

- Makes your system less stable and responsive, prone to divergences and oscillations. Overall it decreases the reliability.

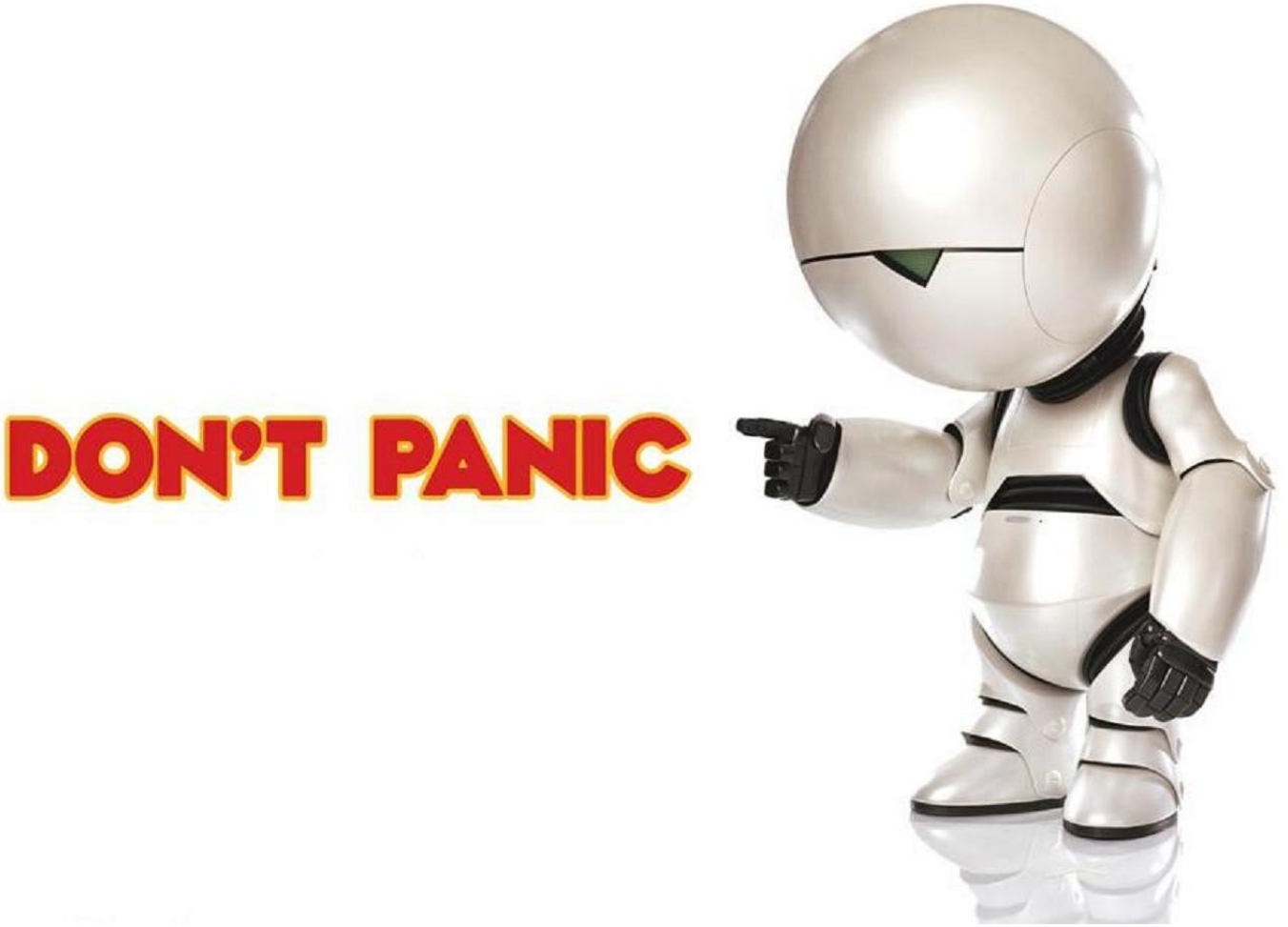
➔Keep it simple, keep under control the free parameters without losing flexibility

- Have you ever heard about SUSY phase-space scans? Do you really want something like that for your DAQ system?

➔Problems require perseverance

- Be careful, a rare little glitch in your DAQ might be the symptom of a major issue with your data

➔In any case, ...


DON'T PANIC