Trigger/DAQ design:
from test beam
to medium size experiments

Roberto Ferrari
Istituto Nazionale di Fisica Nucleare

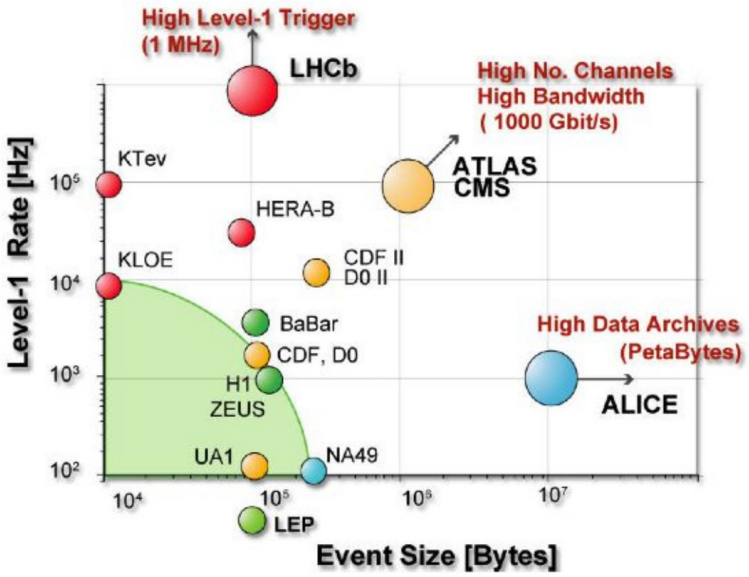
ISOTDAQ 2016
Weizmann Institute of Science
27 January 2016

credit to Sergio Ballestrero
~all material comes from his talk at ISOTDAQ 2015

UNIVERSITY OF JOHANNESBURG

Trigger/DAQ design:
from test beam
to medium size experiments

CERN

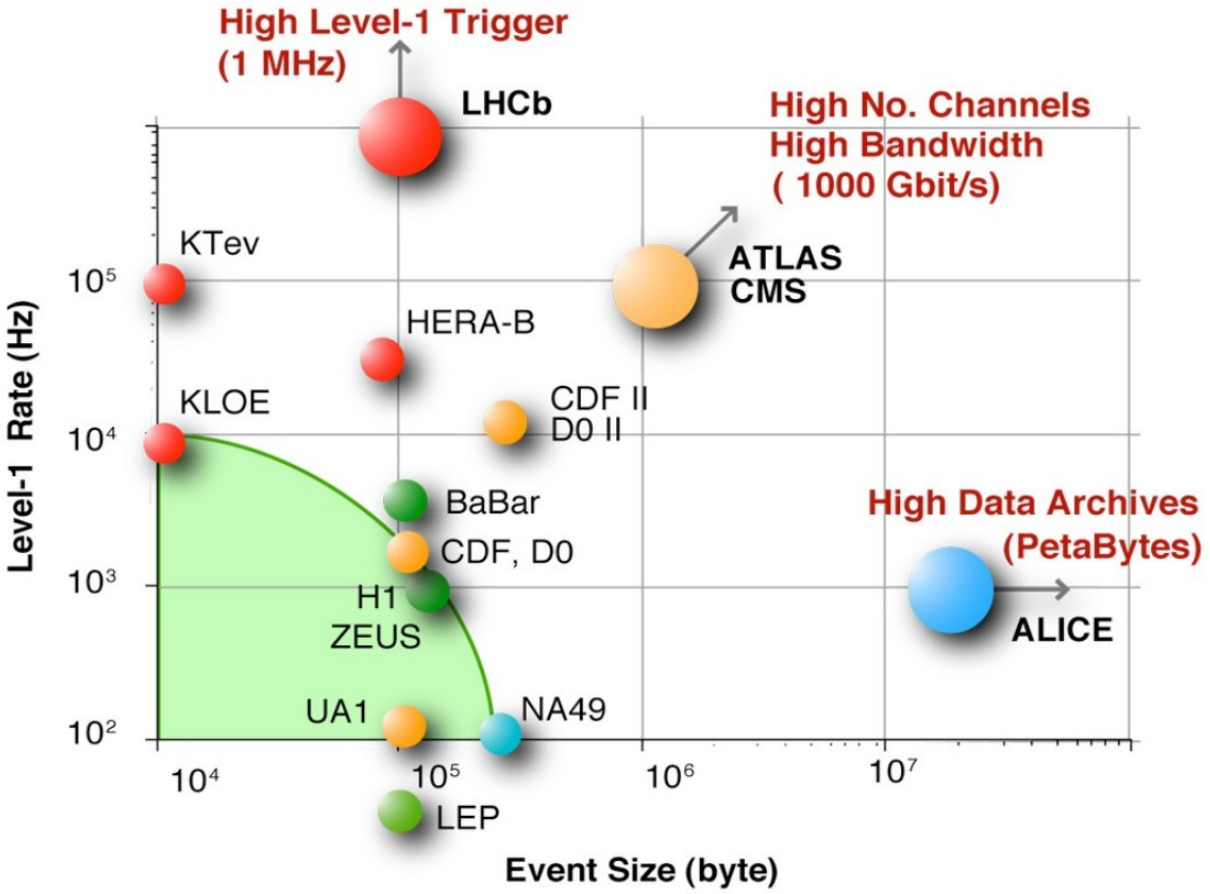


disclaimer

USE THIS MATERIAL AT YOUR OWN RISK

BE AWARE THAT ANY INFORMATION YOU MAY FIND MAY BE INACCURATE, MISLEADING, DANGEROUS, ADDICTIVE, UNETHICAL OR ILLEGAL

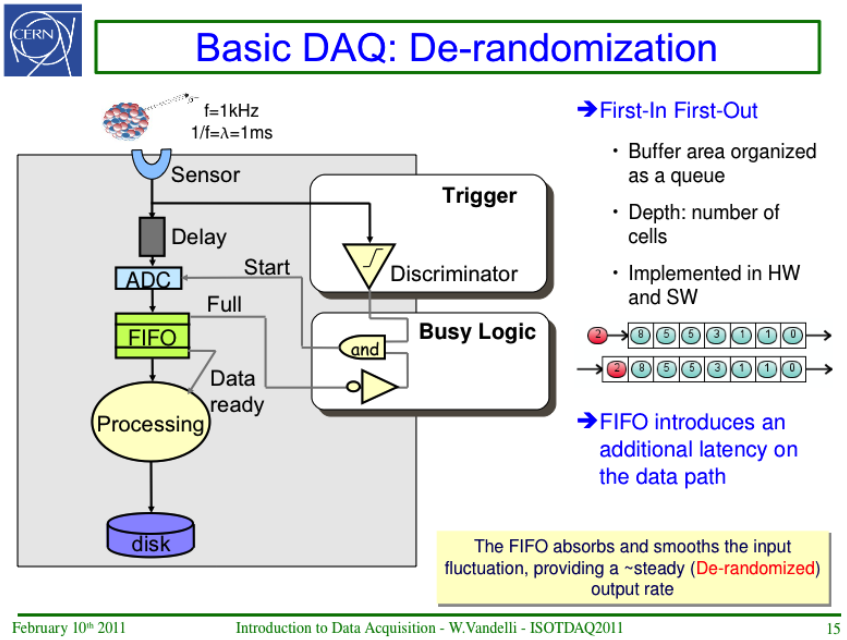
HEP DAQ phase-space



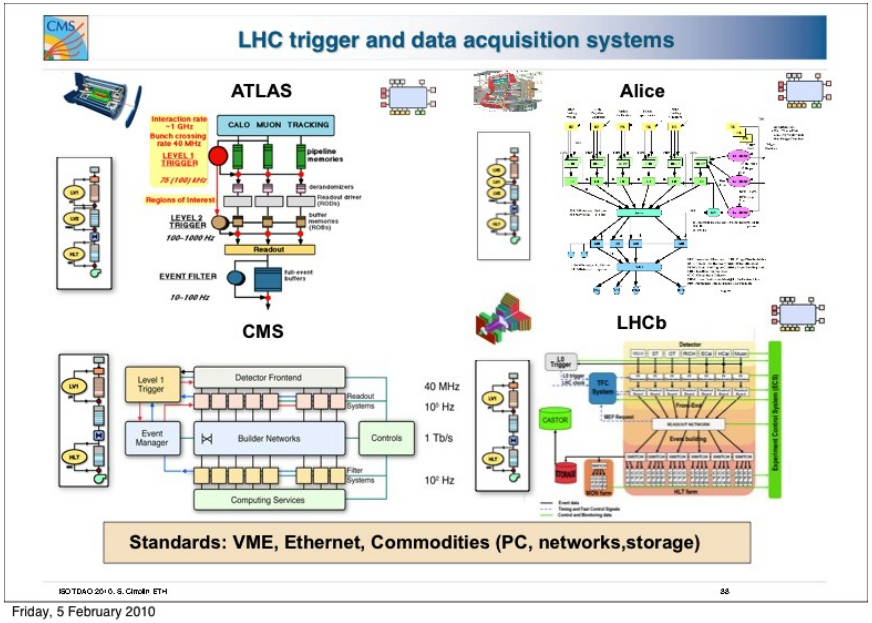
Take care: different issues → different solutions

no single magic solution to all cases

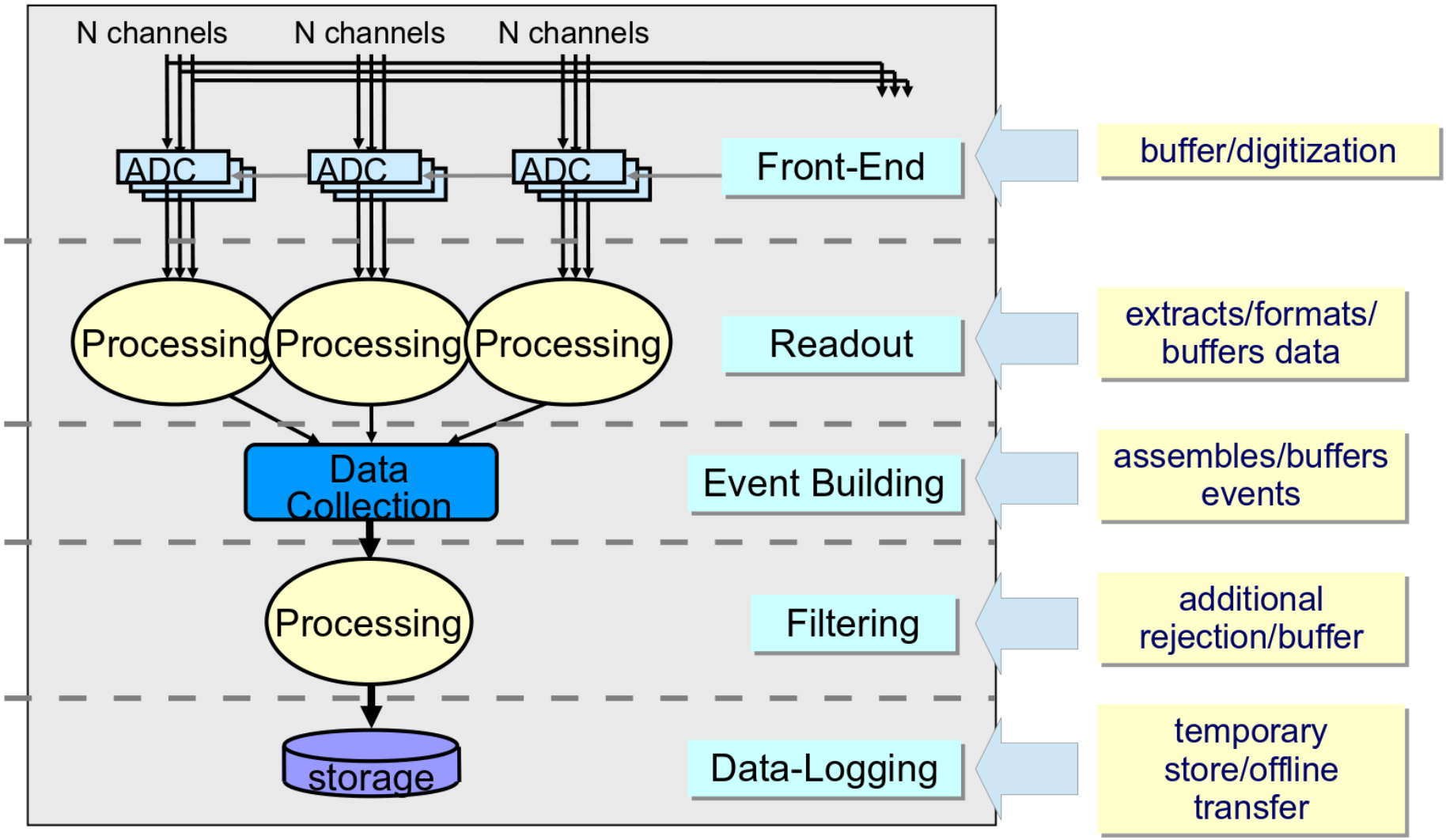
Trying to move …
from here:



to here:



Medium/Large DAQ: constituents



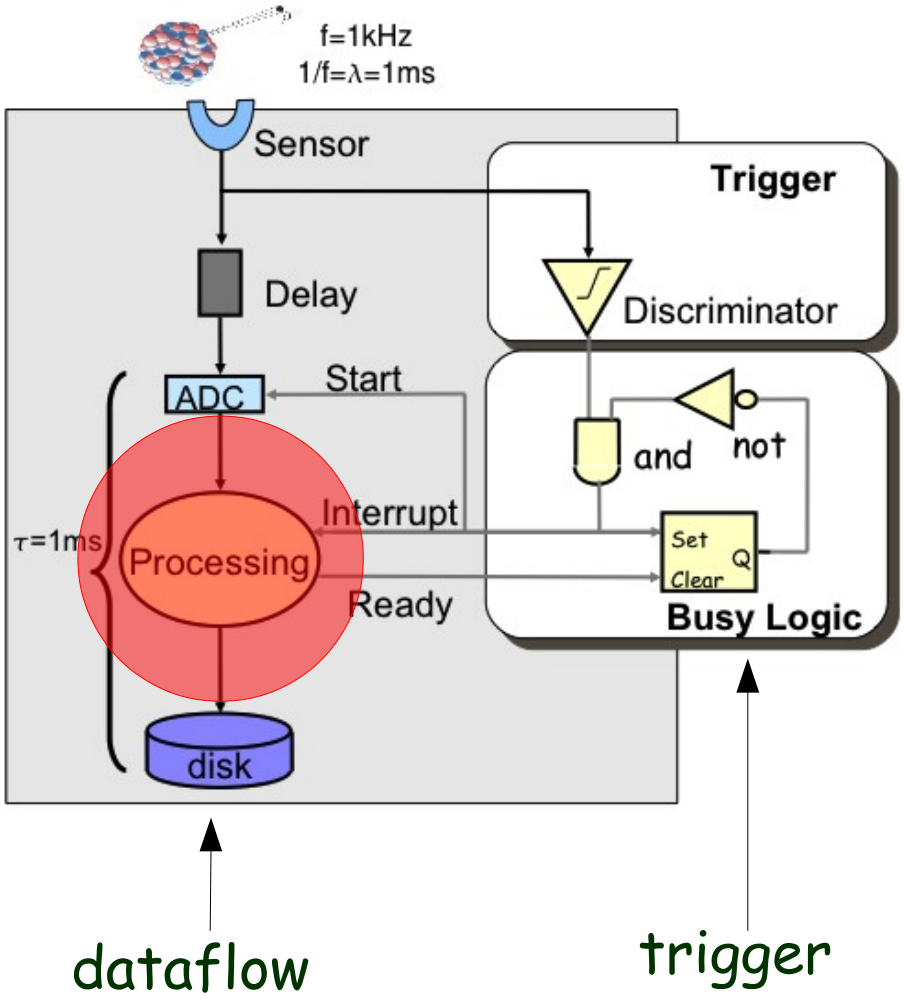
trying to get there in 5 steps …

- Step 1: Increasing the rate
- Step 2: Increasing the sensors
- Step 3: Multiple Front-Ends
- Step 4: Multi-level Trigger
- Step 5: Data-Flow control

step one: increase rate

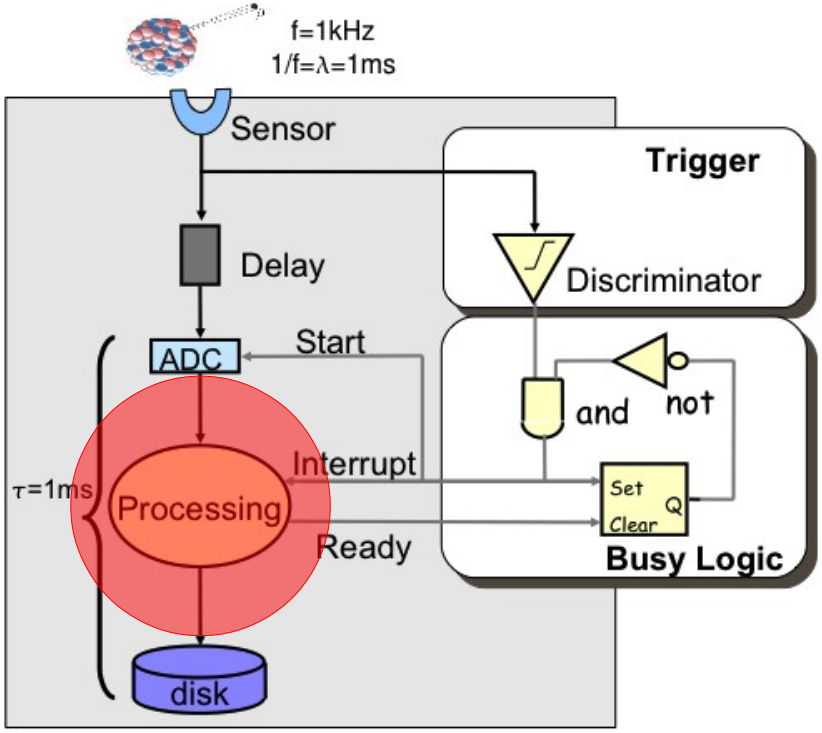
Single-event readout:

- wait for data (poll/irq)
- read ADC
- clear & re-enable ADC
- re-format data
- write to storage



dead time → de-randomise

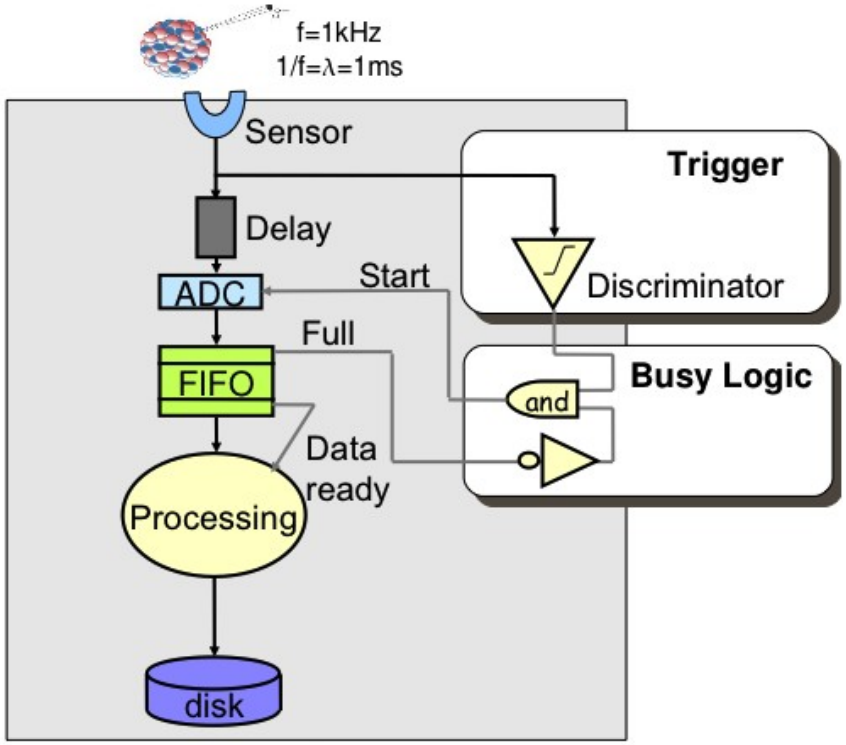
- Processing → bottleneck



Dead time ~ $(1+x)^{-1}$ ~ 50%

[for $x = 1/(f \cdot \tau) \sim 1$]

- Buffering allows to decouple problems



Dead time ~ $(\sum^{0..N} x^j)^{-1}$ ~ $1/(N+1)$

[N = buffer depth]

derandomisation

N-event buffer ... single queue size N:

P_k : % time with k events in buffer

P_N = no space available → dead time

$\sum P_k = 1$ [k=0..N]

rate(j→j+1) = $f \cdot P_j$

rate(j+1→j) = P_{j+1}/τ

stationary condition: $f \cdot P_j = P_{j+1}/\tau \rightarrow P_j = P_{j+1}/(f\tau) = x \cdot P_{j+1}$

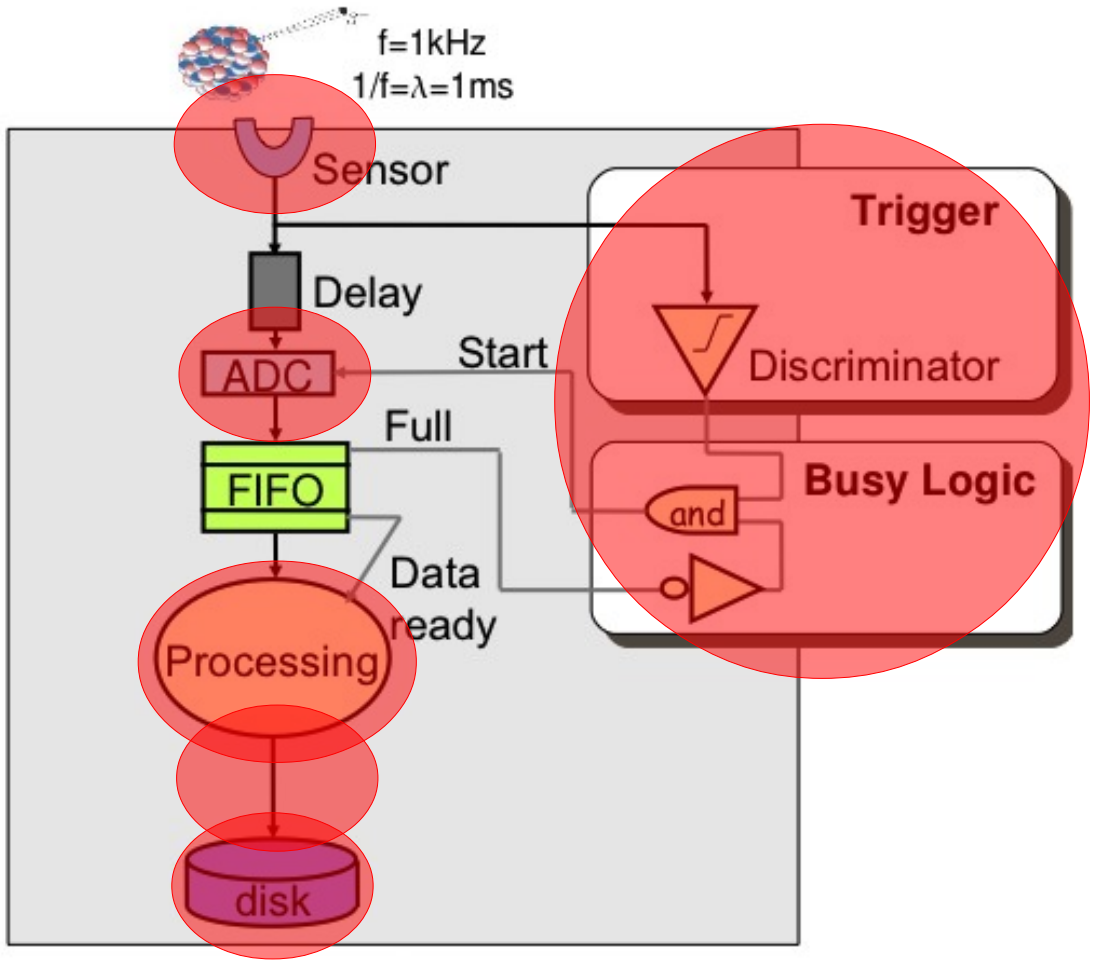
if $x \sim 1 \rightarrow P_j \sim P_{j+1} \rightarrow \sum P_k \sim (N+1) \cdot P_0 = 1 \rightarrow P_0 \sim 1/(N+1)$

→ dead time ~ 1/(N+1)

want ≤ 1% → N ≥ 100

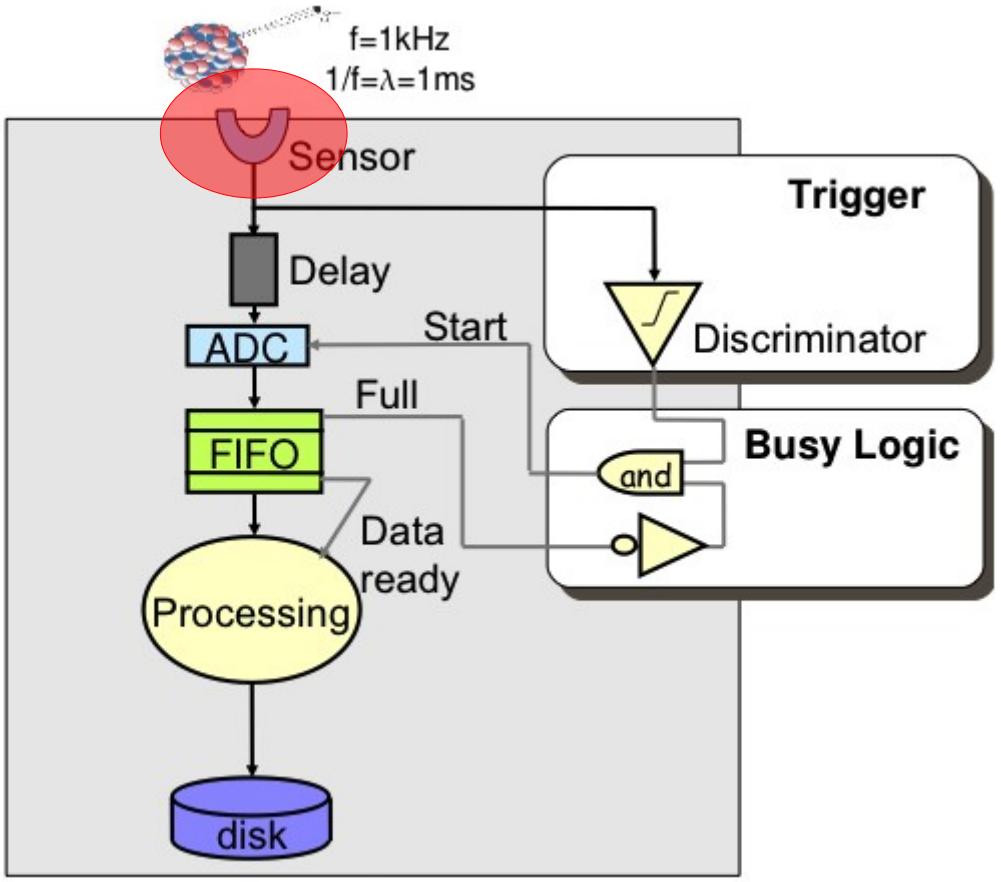
Game over ?

Even in a simple DAQ there are many other possible limits



→ the sensor

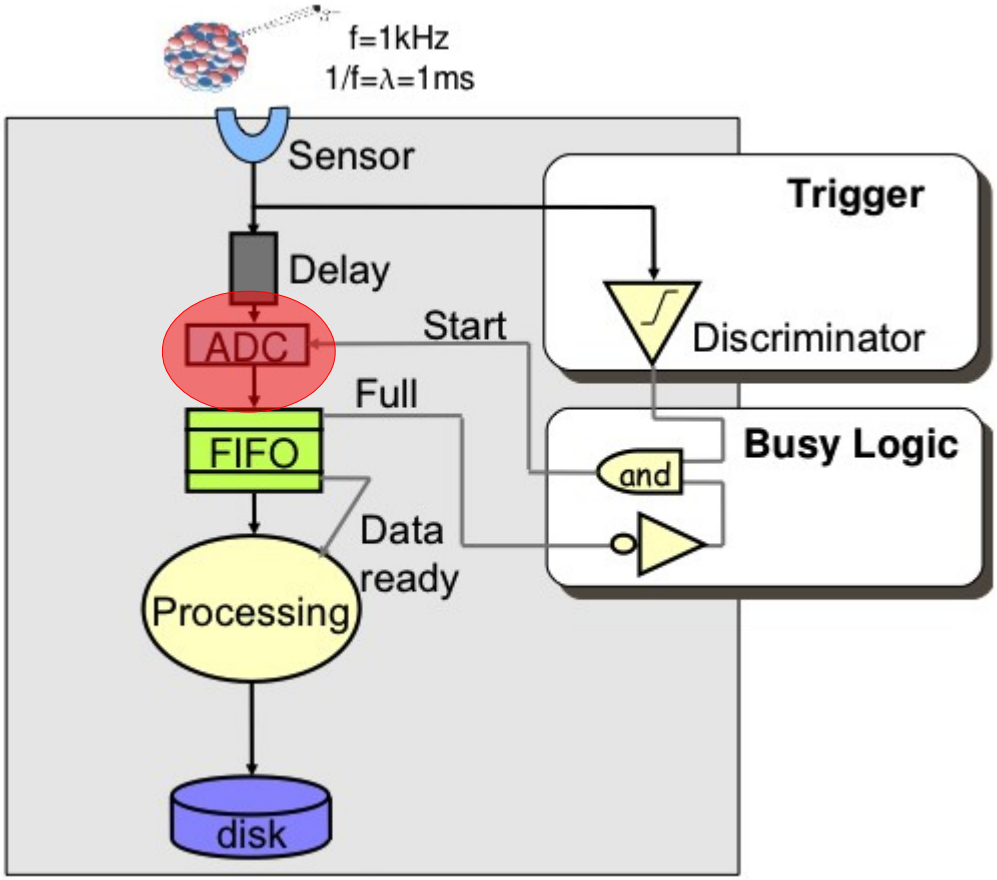
- Sensors are limited by physical processes, e.g.
  - drift times in gases
  - charge collection in Si
- (possibly) choose fast processes
- analog F.E. imposes limits as well
- split the sensors, each gets less rate: "increase granularity"



→ the ADC

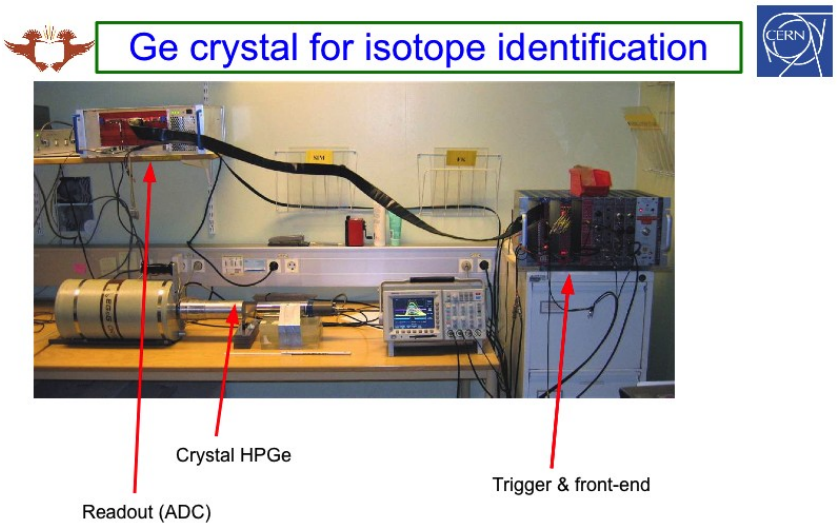
- A/D F.E. is also limited
- Faster ADCs pay the price in precision (# of bits) and power consumption
- Alternatives:
  - analog buffers
- You may need integration (or sampling) over quite some time

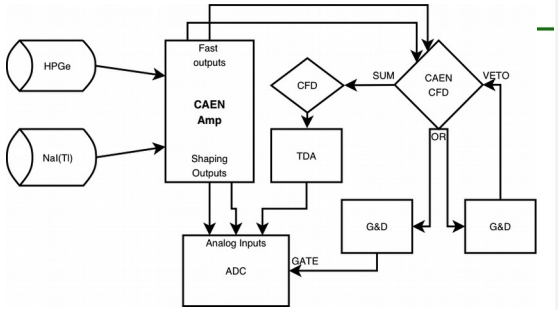
[see Detector Readout and FE lectures]

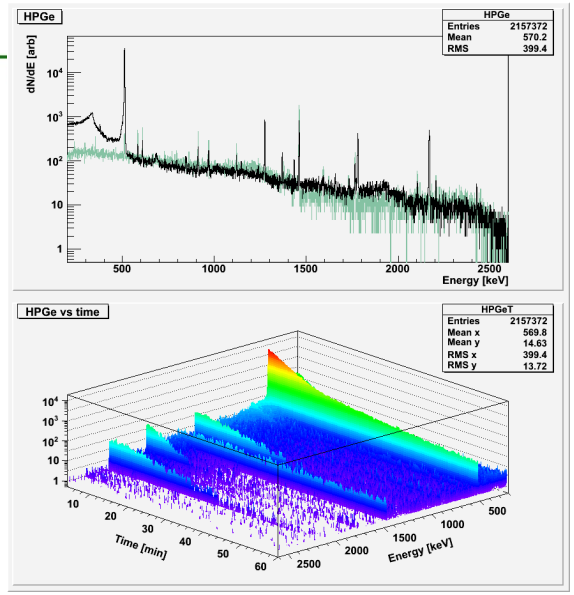


an example

- HPGe + NaI Scintillator
  High res spectroscopy and beta+ decay identification
- minimal trigger with busy logic
- Peak ADC with buffering, zero suppression
- VME SBC with local storage
- Rate limit ~14kHz
  - HPGe signal shaping for charge collection
  - PADC conversion time
- 3x12 bits data size (coincidence in an ADC channel) +32bit ms timestamp
- Root for monitor & storage

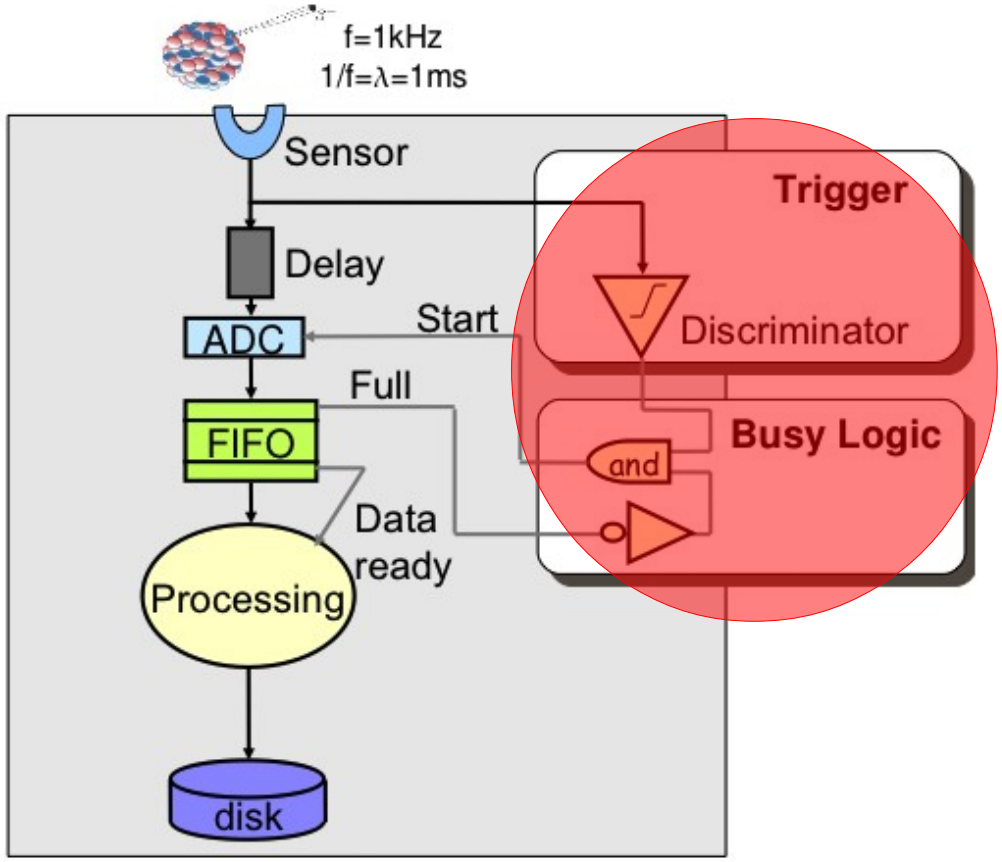






→ the trigger

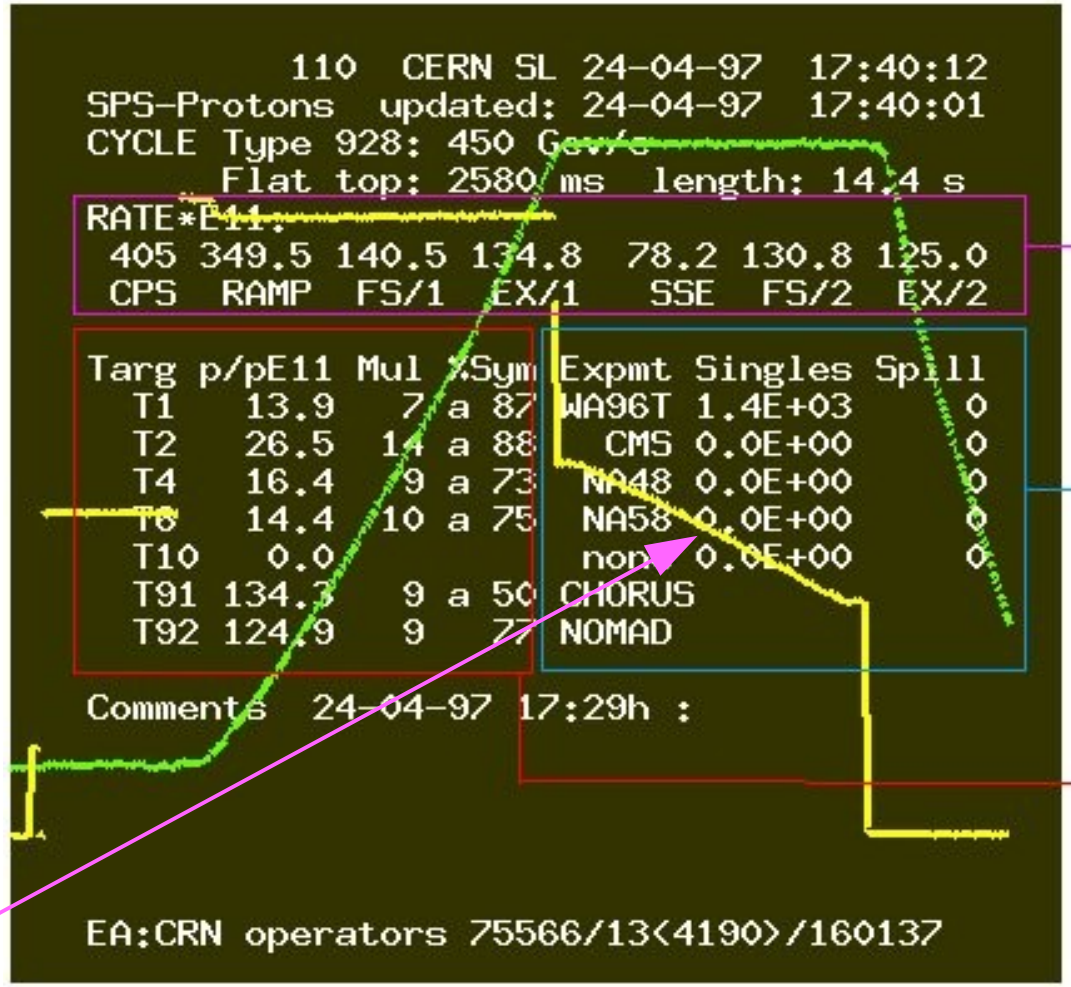
- a simple trigger may be ~fast
- a complex trigger logic may not be [even when all in hw]
- some trigger detectors may be far away / slow → latency
- trigger signal is one: all information must be collected at a single point
  - in one step: too many cables
  - in many steps: delays



→ discrete modules: ~ 5-10 ns delay → tot. latency ≥ 20-30 ns ←

a testbeam case → DREAM

a possible SPS cycle (superCycle) beam: 2.58s / 14.4s (flat top)

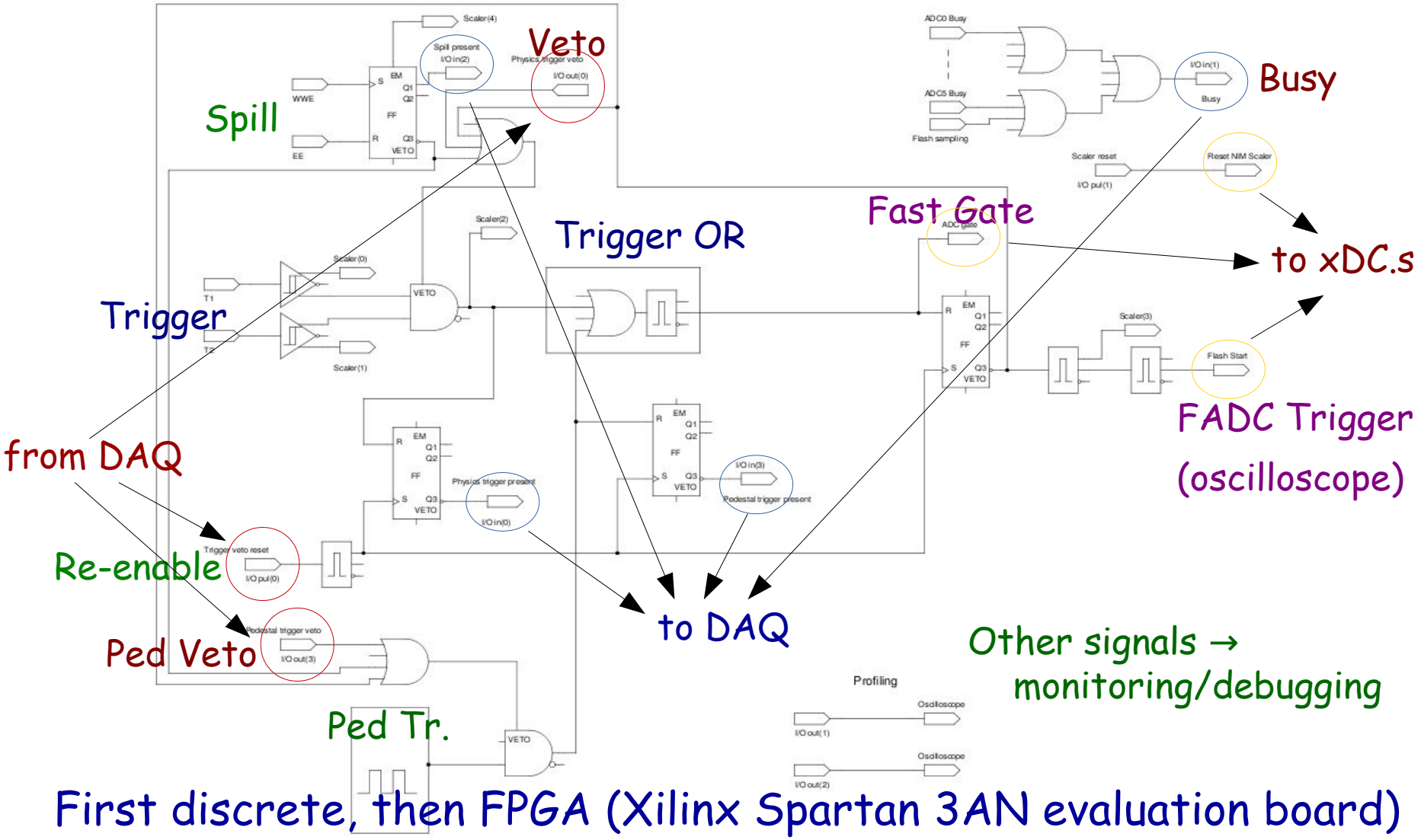


slow extraction

Trigger = $\overline{V} \times T_1 \times T_2$ | ped → easy !

"spill-driven" (asynchronous) trigger

Trigger = !V×T_1×T_2 | ped



First discrete, then FPGA (Xilinx Spartan 3AN evaluation board)

DREAM DAQ

1 PC → 2 VME crates (access via CAEN optical interfaces) + 1 PC → storage
6 x 32 ch xDC.s (x = Q, T : CAEN V792, V862, V775)
1 x 34 ch (CAEN V1742) 5Gs/s Digitizer (single event: ~34x1024x12bit)
1 x 4 ch Tektronix TDS7254B 20 Gs/s oscilloscope
… few VME I/O & discriminator boards

DAQ logic spill-driven (no real time, PC with scientific linux)

in-spill (slow extraction)

a) poll trigger signal … if trigger present:

b) read all VME boards (w/ DMA, whenever possible)

c) format & store on a large buffer (FIFO over RAM)

d) re-enable trigger

out-of-spill

a) read scope (in case) → size is fixed at run start

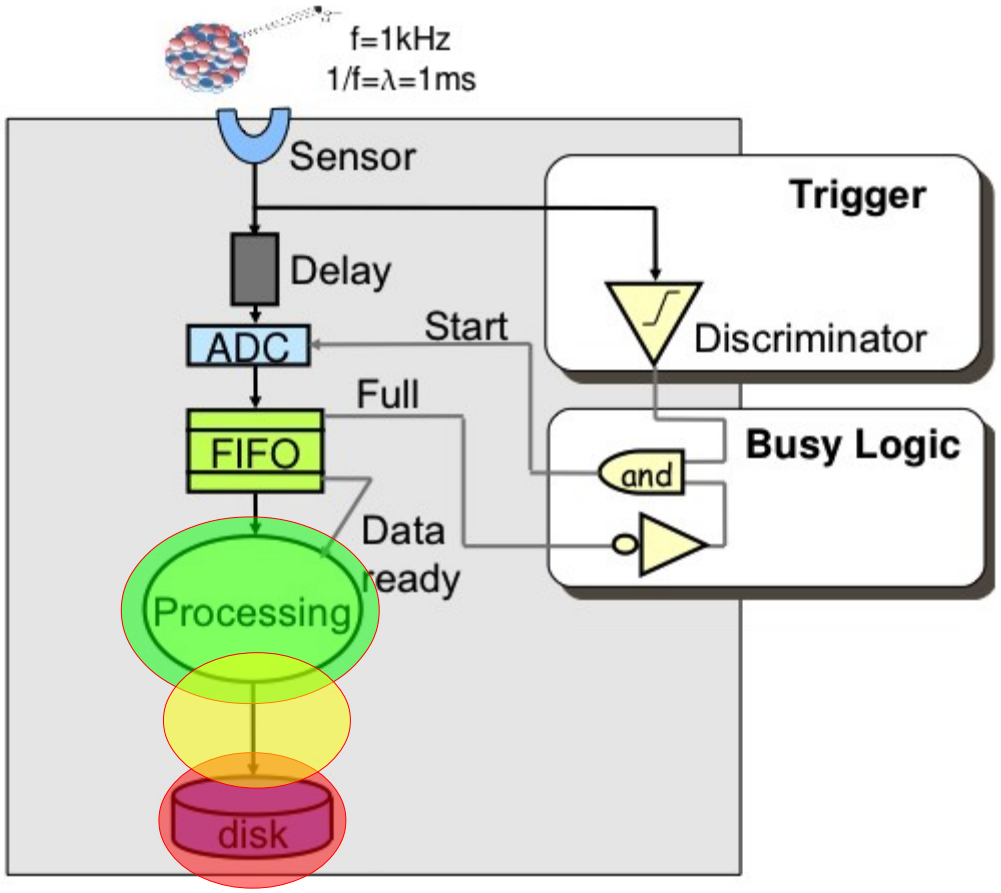
b.1) monitor data (produce root files)

b.2) store on disk files (beam and pedestal files) over network

rate ~ O(1 kHz)

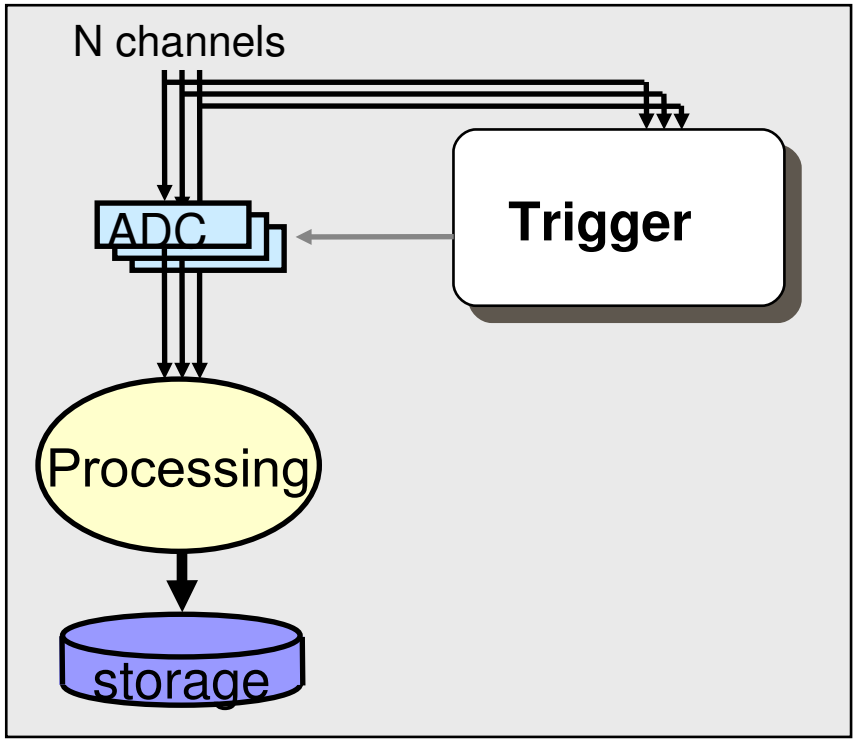
→ the dataflow

- Data Processing may be ~ easy and scalable
- Data Transport may not be easy
- Final storage is expensive (and at some point not easy either) → can't store all data you may acquire

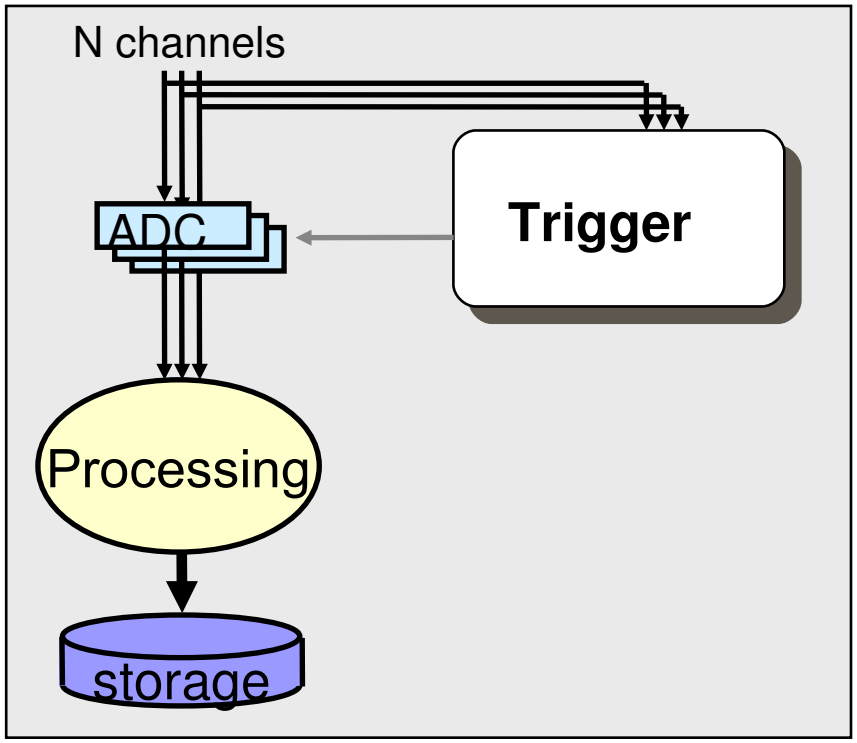


step two: increase # of sensors

- More granularity at the physical level
- Multiple channels (usually with FIFOs)
- Single, all-HW trigger
- Single processing unit
- Single I/O

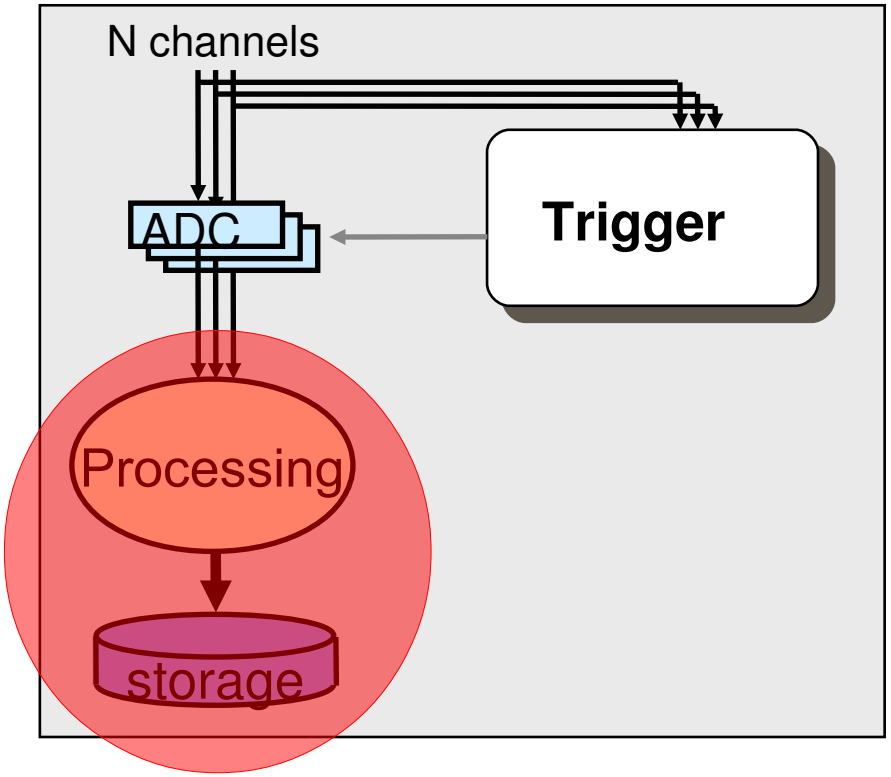


multi-channels, single PU



- common architecture in test beams and small experiments
- often rate limited by (interesting) physics itself, not TDAQ system
- or by the sensors

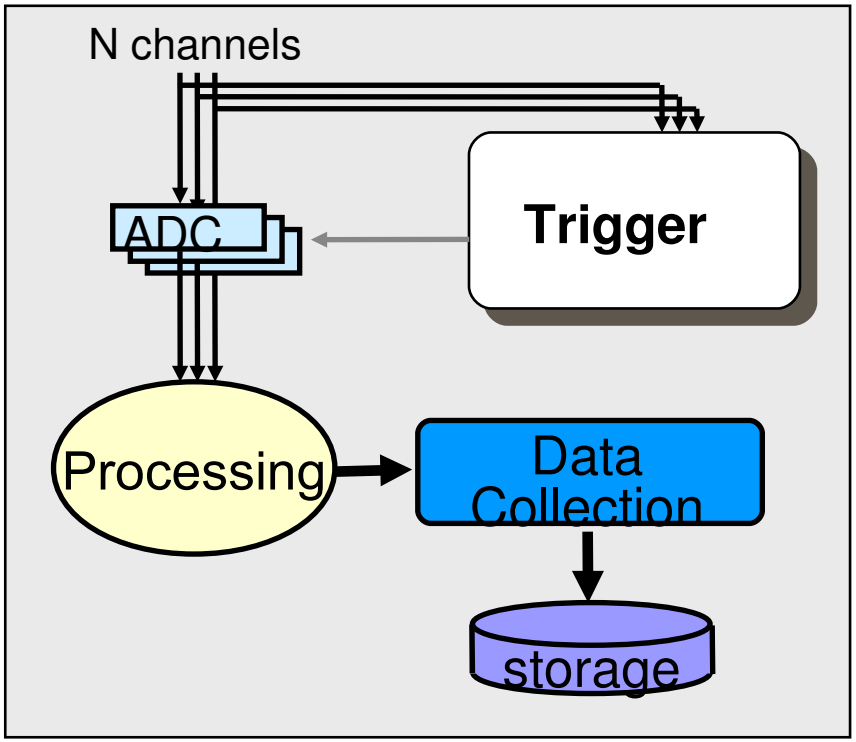
bottlenecks: PU and storage



- a single Processing Unit can be a limit
  - collect / reformat / compress data can be heavy
  - simultaneously writing storage
- final storage too:
  - VME up to 50MB/s
    -> 1TB in 6h
    too many disks in a week!

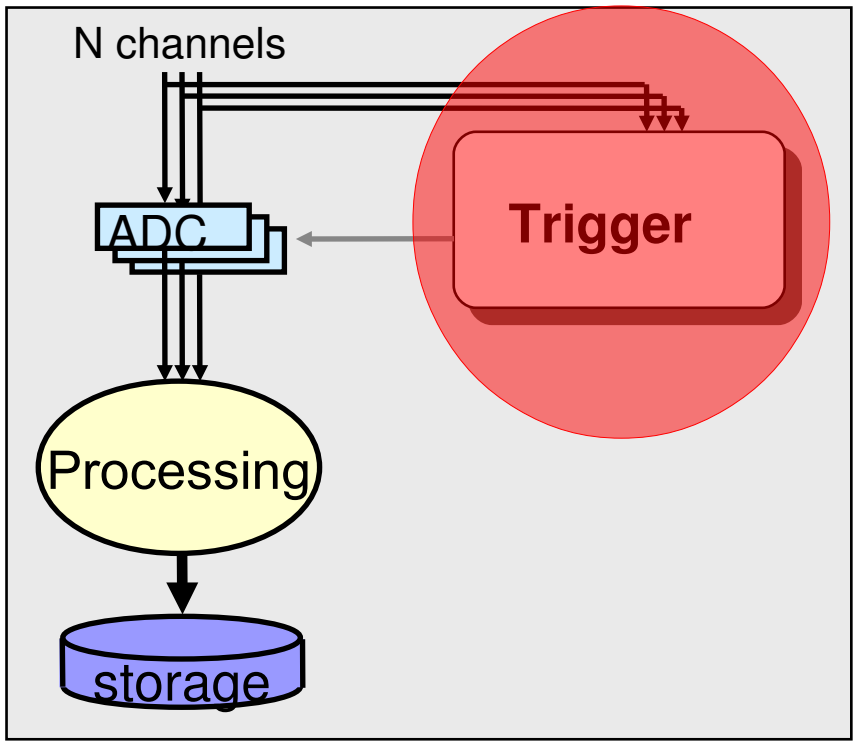
Laptop SATA disk: 54MB/s; USB2: ~30MB/s

→ decouple storage from PU



- data transfer data → dedicated "Data Collection" unit to format, compress and store
- more room for smarter processing or decreased dead time on non-buffered ADCs

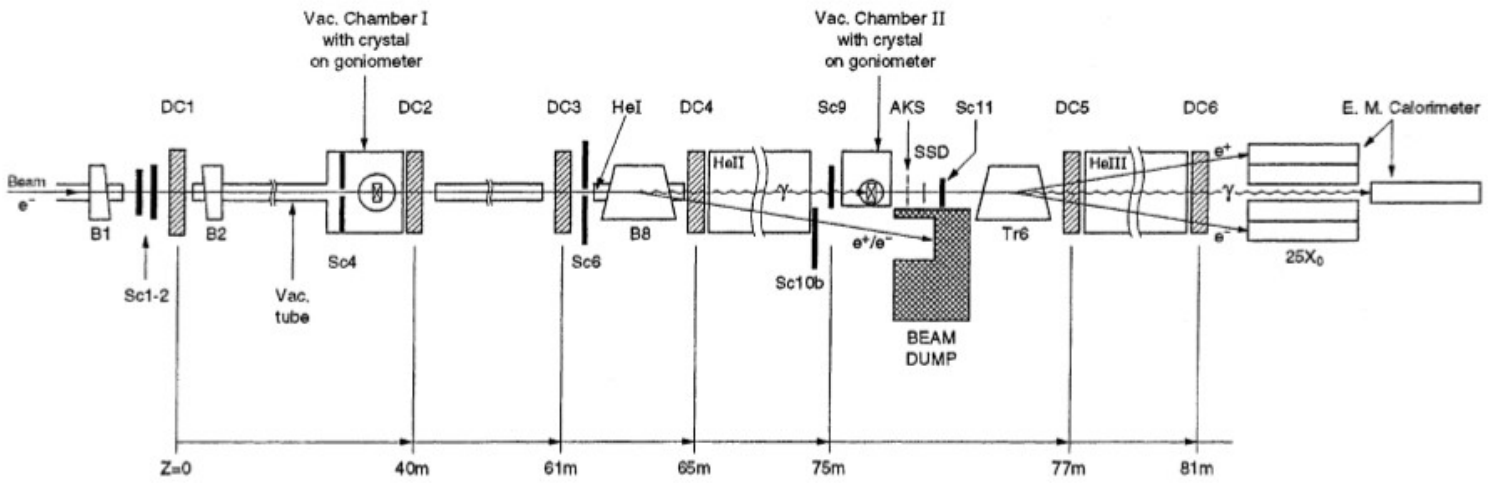
bottlenecks: trigger



- to reduce data rates
  (to avoid storage issues)
  → non-trivial trigger
- complexity may already hit manageability limits for discrete logic (latency!)
- integrated, programmable logic came to rescue (FPGA)

  → latency may go down to O(few ns)

another example: NA43/63

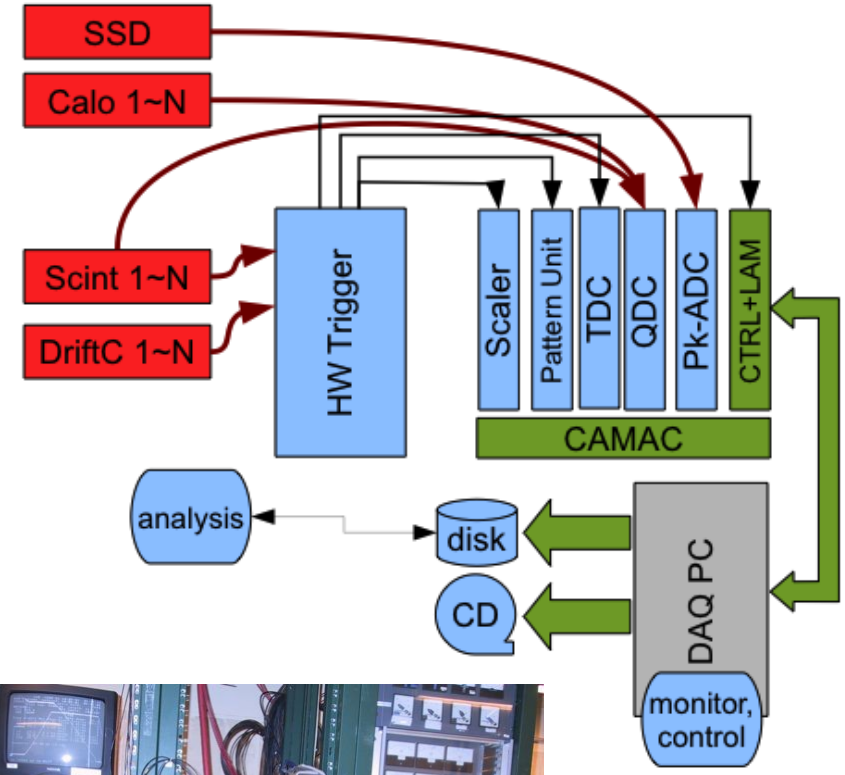


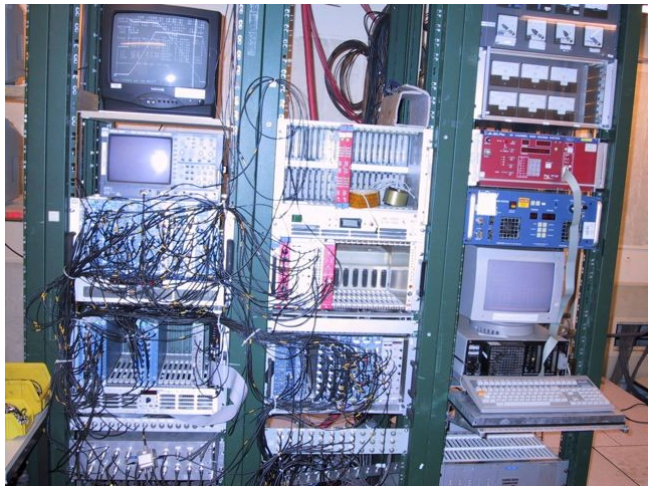
- Radiation processes: coherent emission in crystals and structured targets, LPM suppression...
- 80~120 GeV e- from CERN SPS slow extraction
- 2s spill every 13.5s

- Needs very high angular resolution
- Long baseline + high-res, low material detectors
  → drift Chambers
- 10 kHz limit on beam for radiation damage
- results in typical 2~3 kHz physics trigger

NA43/63

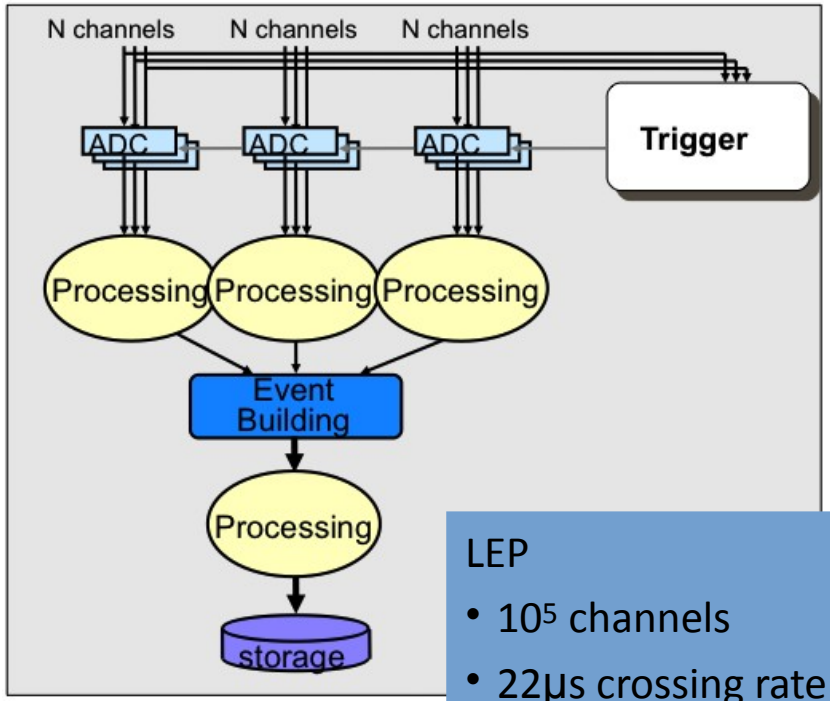
- 30~40 TDC, 6~16 QDC, 0~2 PADC
  (depends on measurement)
- CAMAC bus
  1MB/s, no buffers, no Z.S.
- single PC readout
- NIM logic trigger
  (FPGA since 2009)
  - pileup rejection
  - fixed deadtime





step three: multiple PU (SBC)

- e.g.: CERN LEP experiments
- complex detectors, moderate trigger rate, very little background
- little pileup, limited channel occupancy
- simpler, slow gas-based main trackers



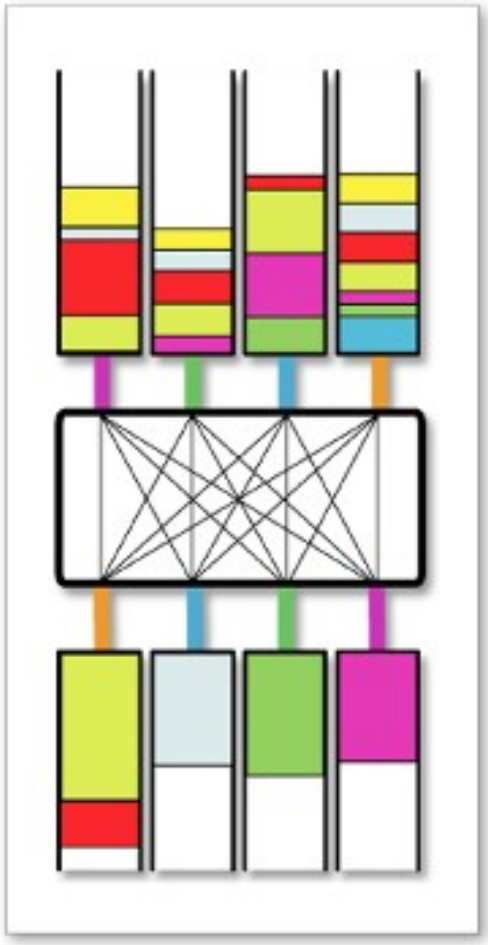
LEP
- 10^5 channels
- 22μs crossing rate
  - no event overlap
- single interaction

→ event building

- Event "fragments" in detector/sector-specific pipeline
- keep track of which event they belong to

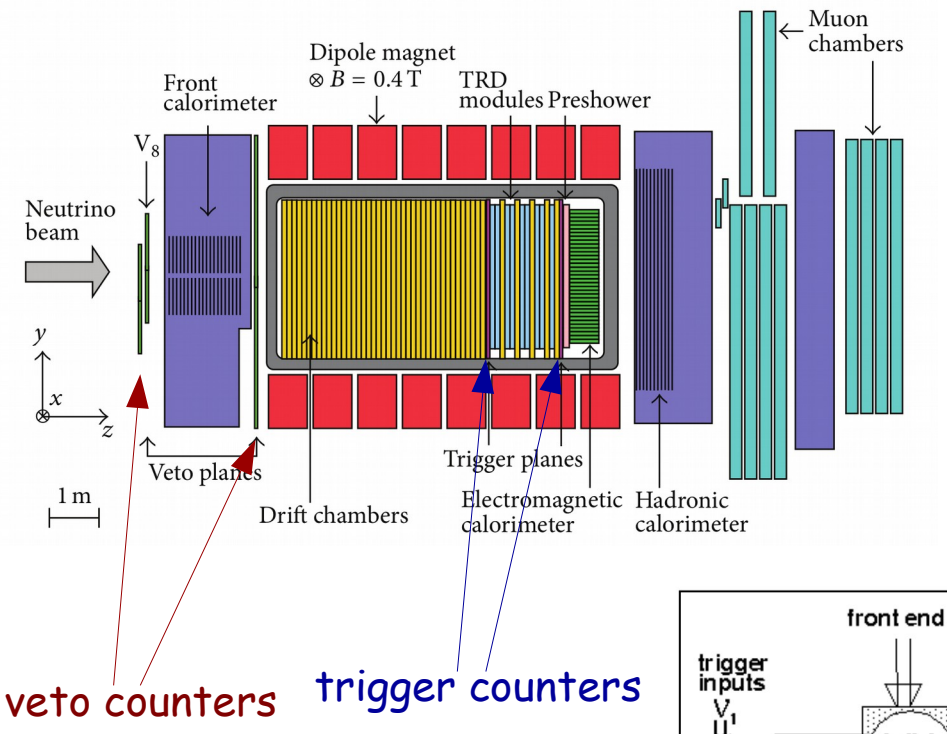
  w/ timestamp or

  w/ L1 trigger #
- gather every fragment to single location
- synchronous/asynchronous



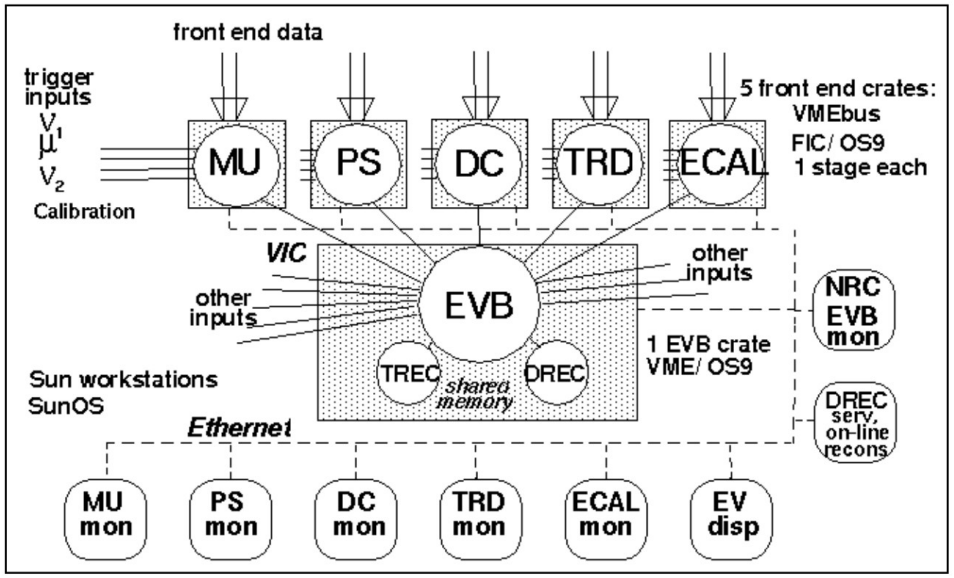
see DAQ Software lecture

NOMAD



- Search for $\nu_\mu \rightarrow \nu_\tau$ oscillations at the CERN WB neutrino facility
- 2.4×2.4 m^2 fiducial (beam) area
- two 4ms-spills with 1.8×10^{13} P.o.T. each
- a (2s) slow-extraction spill
- cycle length of 14.4 s

DAQ layout



NOMAD DAQ

- ~30(?) (64 or 96 channel) Fastbus xDC boards [x = Q, P, T]
- Typically:
  - ~15 evts each 4ms spill (neutrino triggers)
  - ~60 evts each 2s-spill (muon triggers)
  - 256-event calibration cycles off-spill (calibration triggers)
  - On spill(cycle): on-board buffering of up to 256 events (no way to read event-by-event)
  - End of spill(cycle): block transfer to 5 VME PU.s (motorola 68040 FIC8234 board, OS9 real-time system)
  - Event building and storage on another VME PU
  - Monitoring and control on SunOs/Solaris workstations

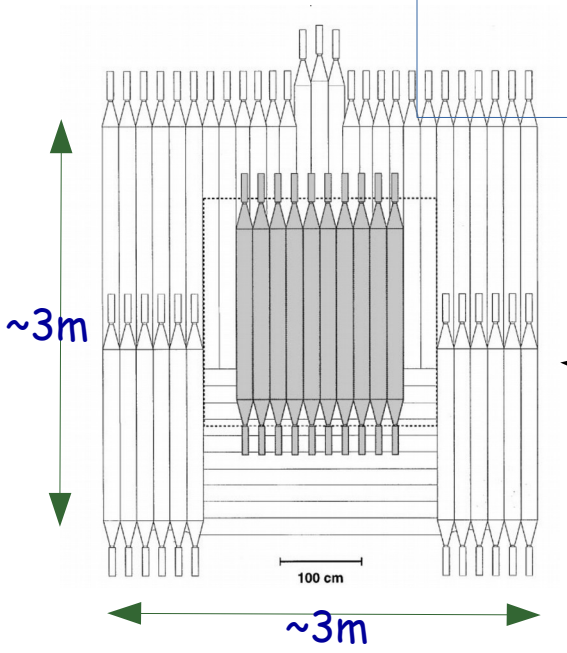
→ on-board buffering

→ data processing is done off-beam (once more)

Triggering once more ...

menu for NOMAD:

ν-spill triggers	μ-spill triggers
$\overline{V} \times T_1 \times T_2$	$V \times T_1 \times T_2$
$\overline{V_8} \times FCAL$	$V_8 \times T_2$
$\overline{V_8} \times FCAL' \times T_1 \times T_2$	$V_8 \times T_1$
$\overline{T_1 \times T_2} \times ECAL, \overline{V_8} \times ECAL$	$V_8 \times T_1 \times T_2 \times FCAL'$
RANDOM	$V \times T_1 \times T_2 \times ECAL$

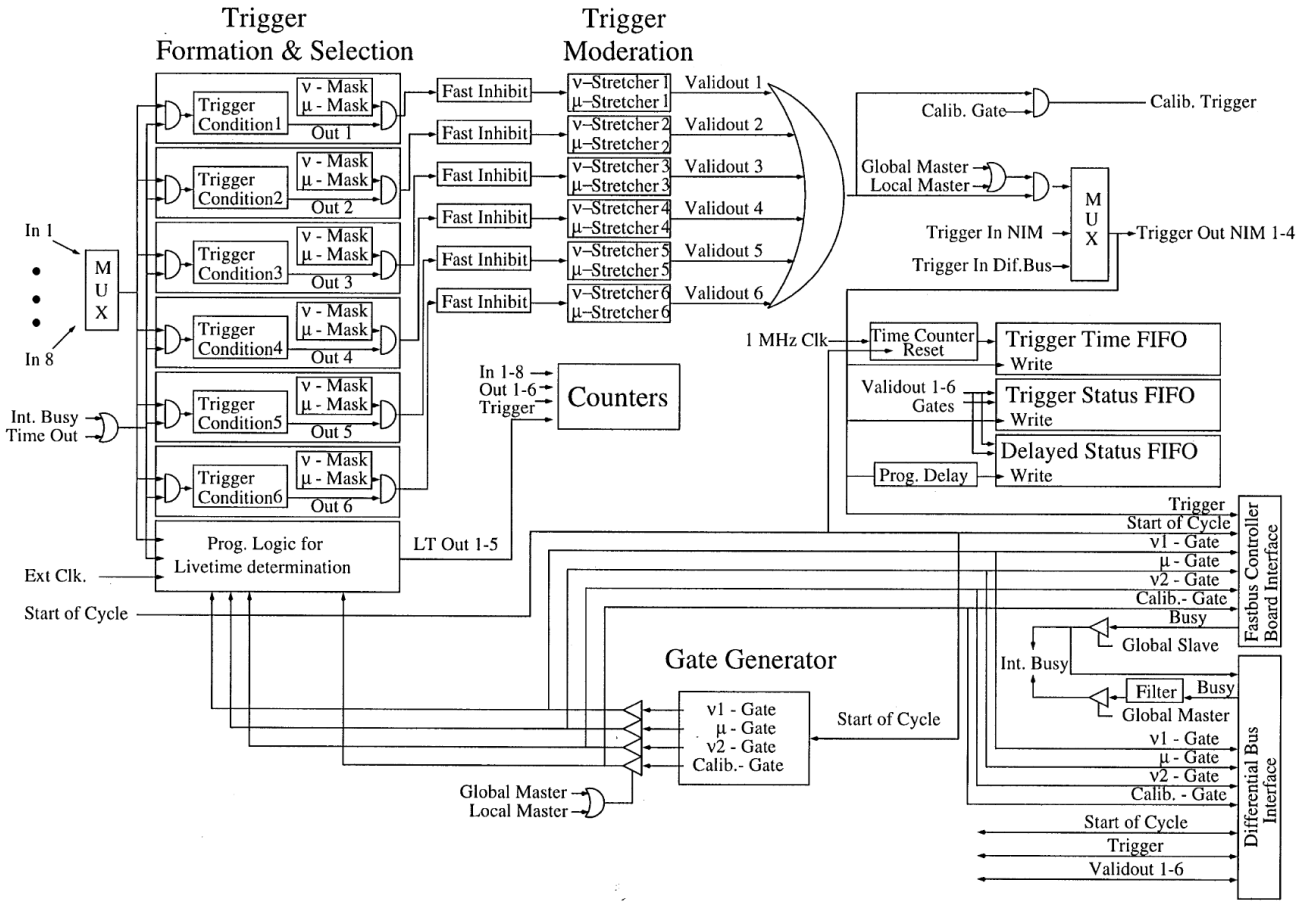


veto counters (central shaded area is V8)

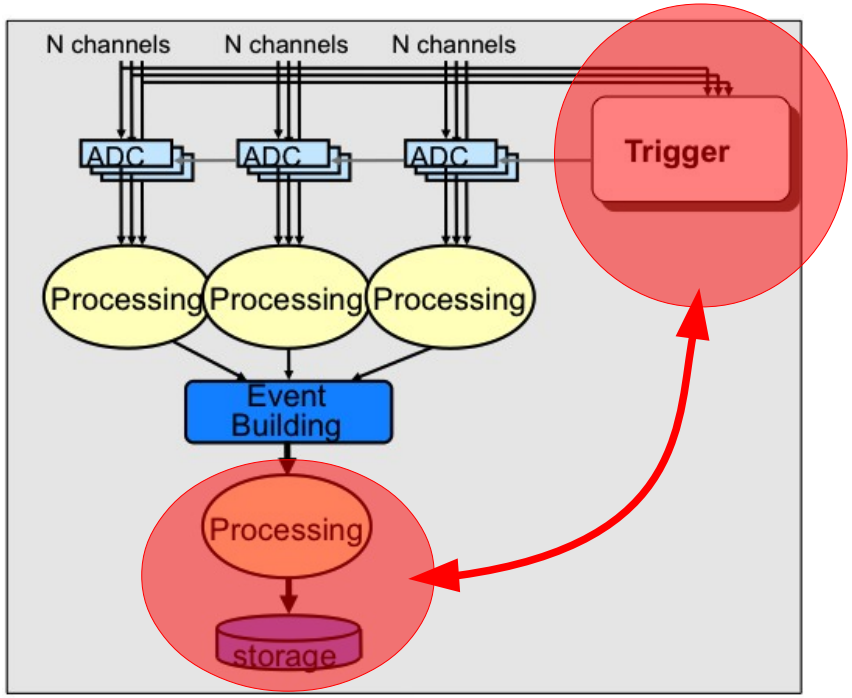
→ FPGA.s at work

MOdular TRIgger for NOmad (MOTRINO):

6 VME boards providing local and global trigger generation and propagation



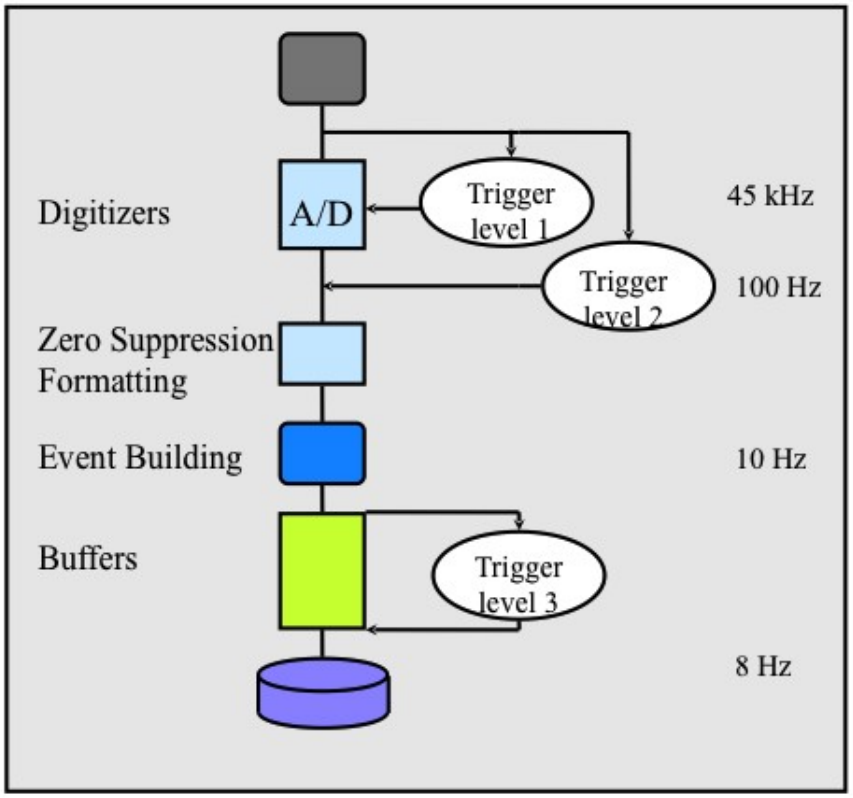
bottlenecks ?



- trigger complexity ↔ storage
- single HW trigger not sufficient to reduce rate
- add L2 Trigger
- add HLT

step four: multi-level trigger

Typical Trigger / DAQ structure at LEP



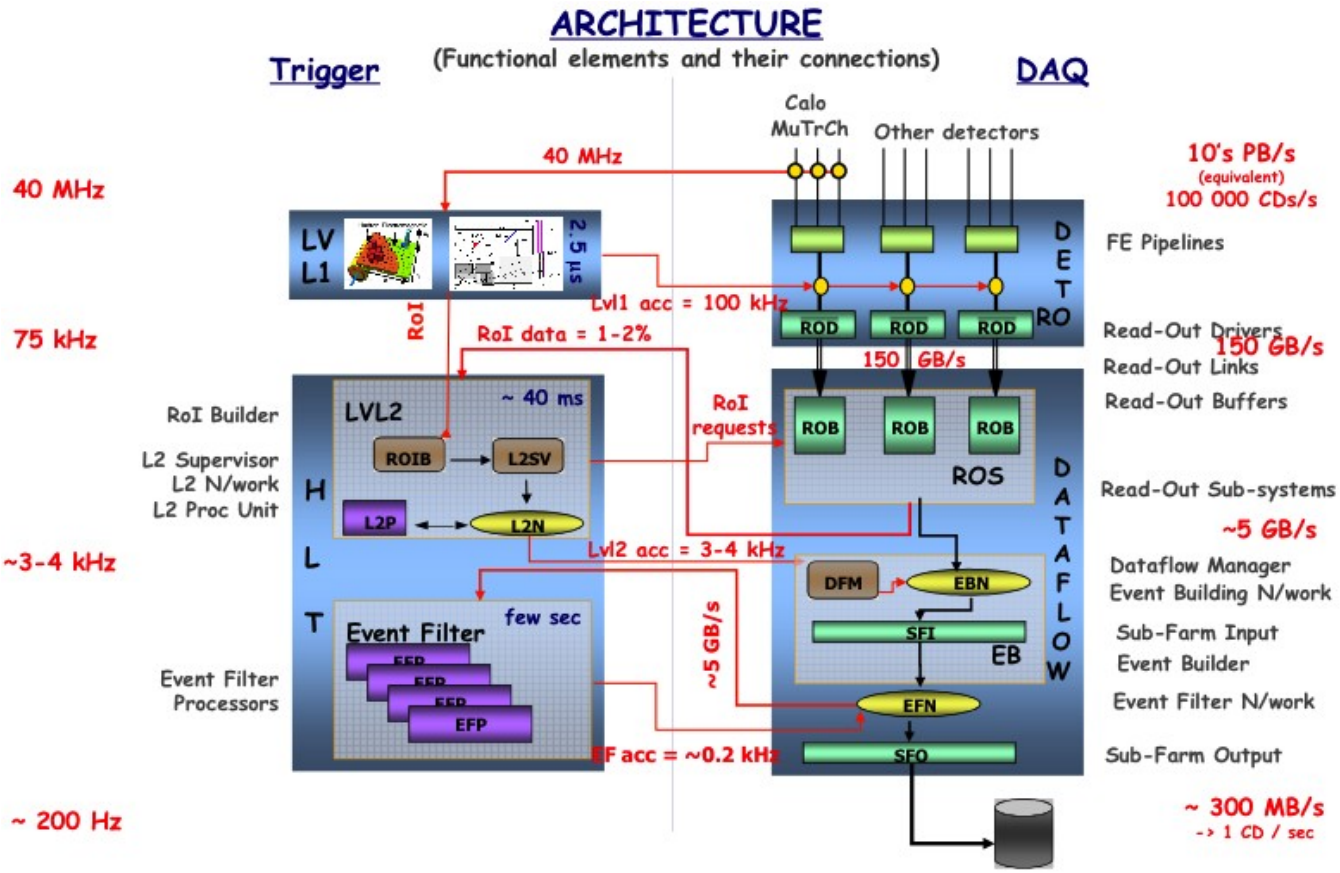
- more complex filters
- → slower
- → applied later in the chain

see Trigger lectures

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

ATLAS!

LHC
- 10^7 channels
- 25ns crossing rate
  - high event overlap
- 20 interactions
- L1 ~10^5 Hz
- L2 ~10^3 Hz
- L3 ~10^2 Hz
- 1MB/ev → 100MB/s



ATLAS T&DAQ Why & How, L. Mapelli @ISOTDAQ 2010

LHC (collider) → sinchronous

… nevertheless, high luminosity & high cross sections → high rate, high-pileup, large events:

➔ most events uninteresting

➔ good events (triggers) arrive uncorrelated (unpredictable)

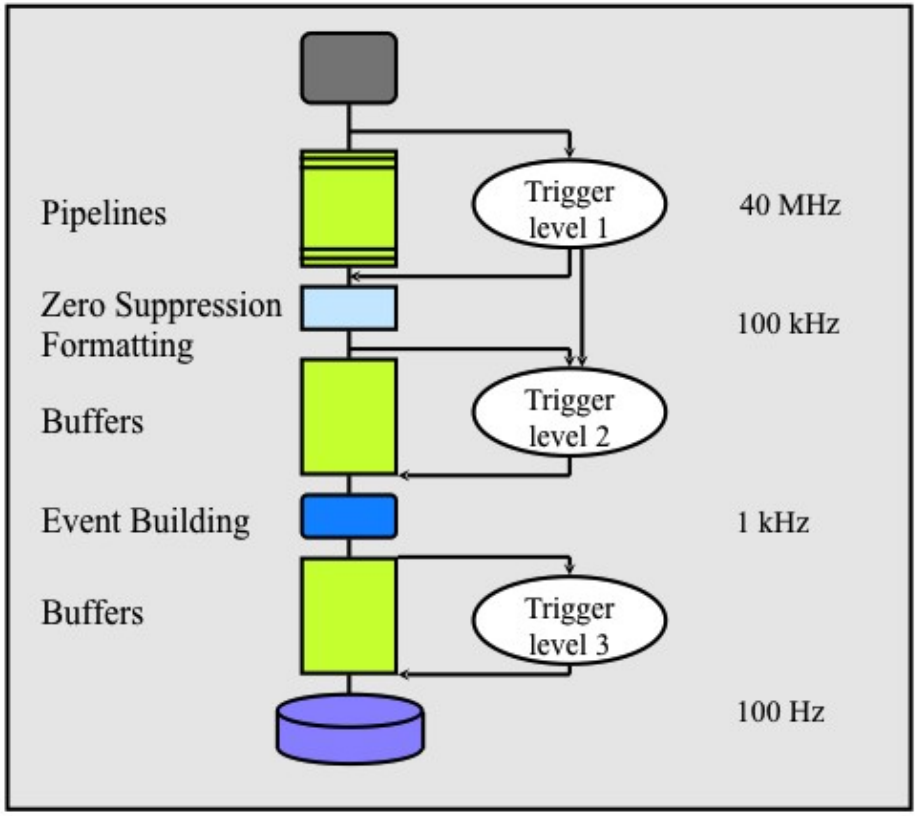
➔ de-randomization is still needed

→ dataflow is an issue

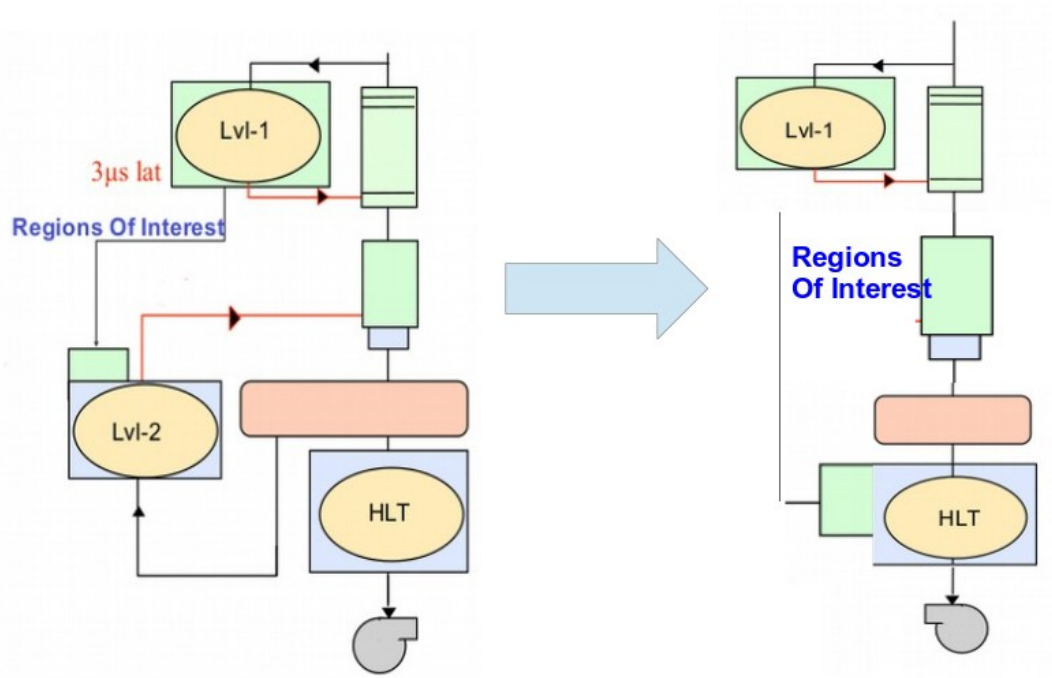
ATLAS run-1 architecture

- Still 3-level trigger
- buffers everywhere
- L2 on CPU, not HW, but limited to ROIs
- L3 using offline algorithms
- "economical" design: the least CPU and network for the job

see "TDAQ for LHC" lecture



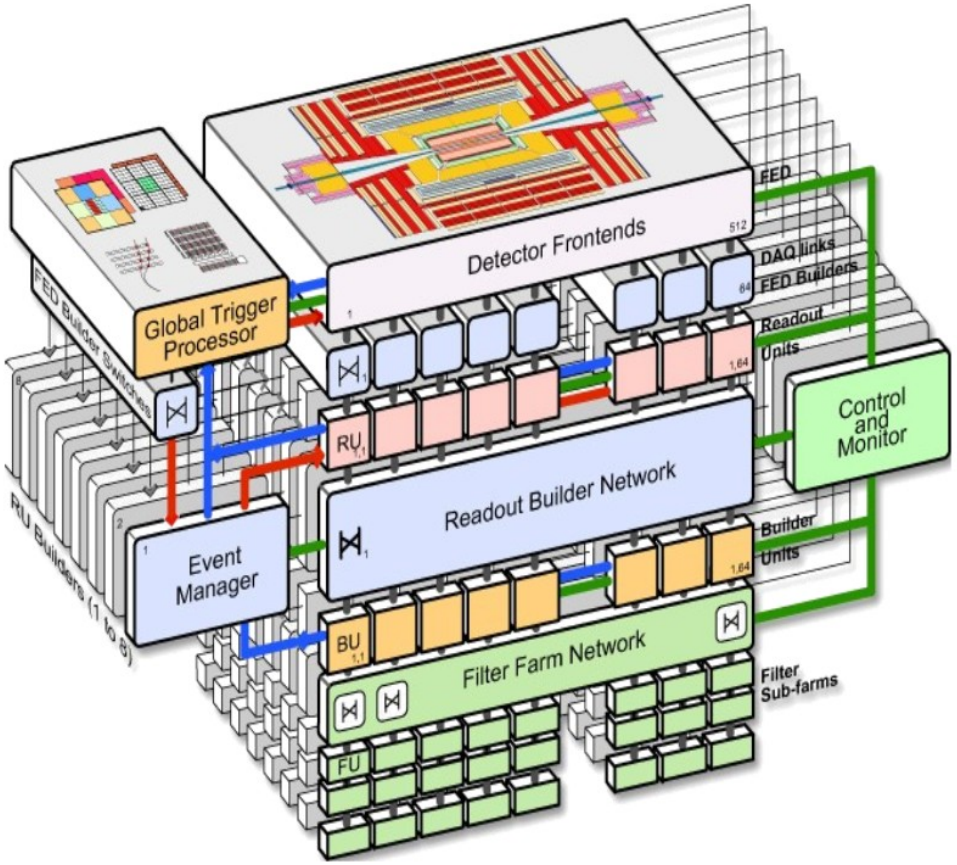
ATLAS run-2 architecture

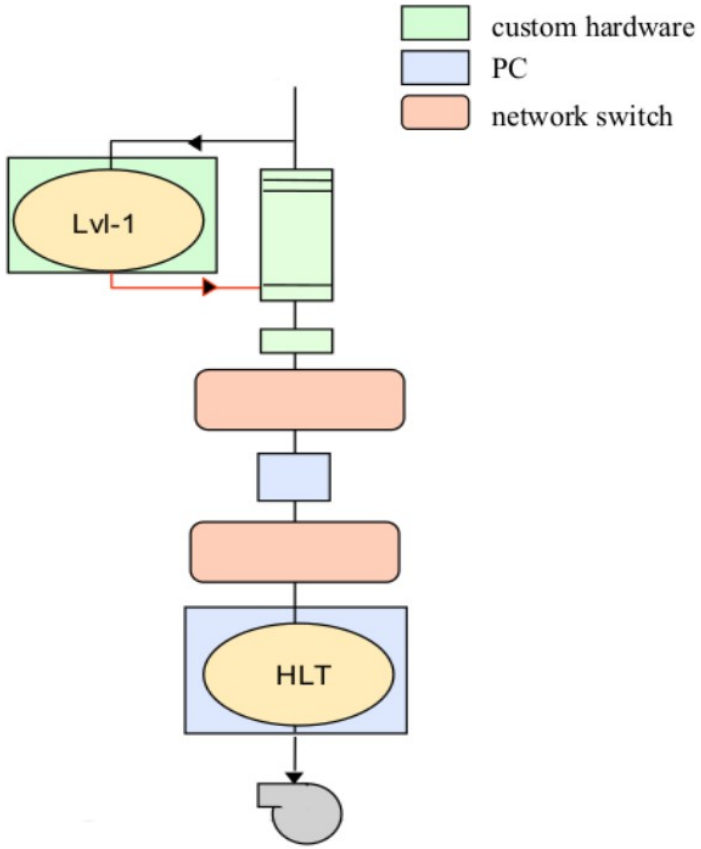


→ Merge L2 and L3 into a single HLT farm

- preserve Region of Interest but dilute the farm separation and fragmentation
- increase flexibly, computing power efficiency

CMS!



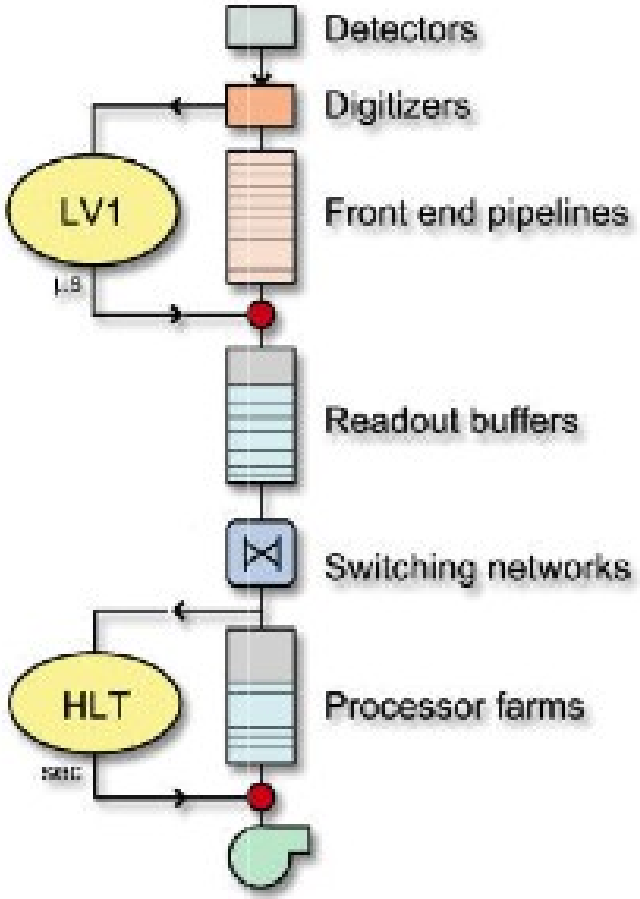


CMS TDAQ Design - S. Cittolin @ISOTDAQ 2010

CMS architecture

- Only two trigger levels
- Intermediate event building step (RB)
- larger network switching

see "TDAQ for LHC" lecture



- upgrade: no architectural changes but:
  - all network technologies replaced
    - Myrinet → Ethernet
    - Ethernet → Infiniband
  - file-based event distribution in the farm
    - full decoupling between DAQ and HLT

Evolution for LHC Run 2

ATLAS:

more like CMS

… still using "L2" ROI, but as first step of a unified L2/EB/HLT process

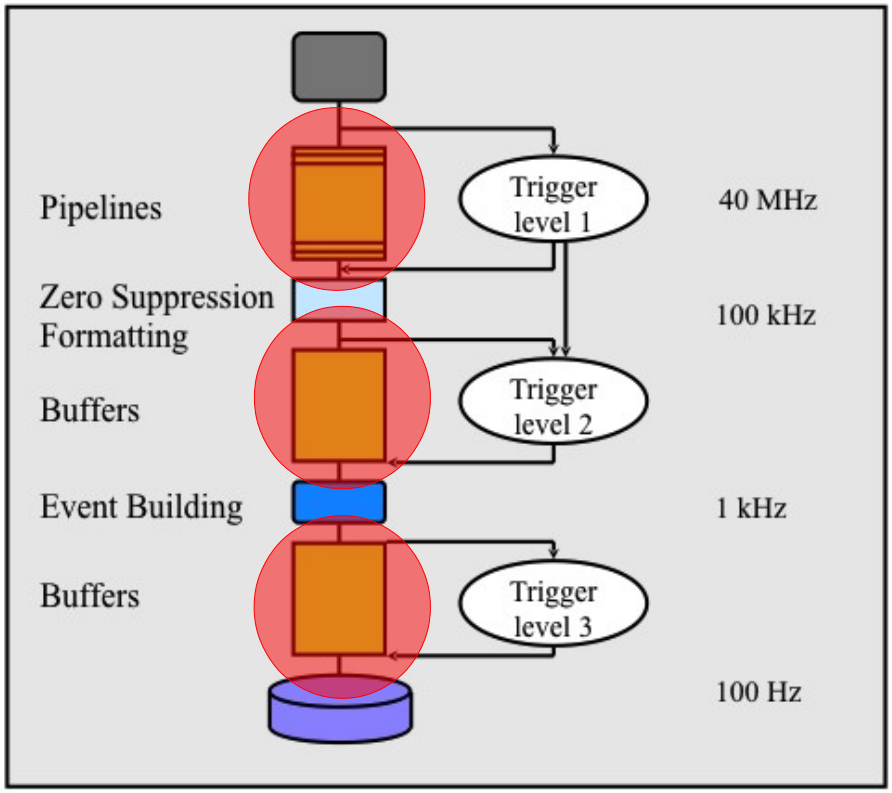
CMS:

more like ATLAS

… still doing full EB, but analyse ROI first

DAQ@LHC Joint Workshop 2013 :
http://indico.cern.ch/conferenceOtherViews.py?view=standard&confId=217480

step five: dataflow control



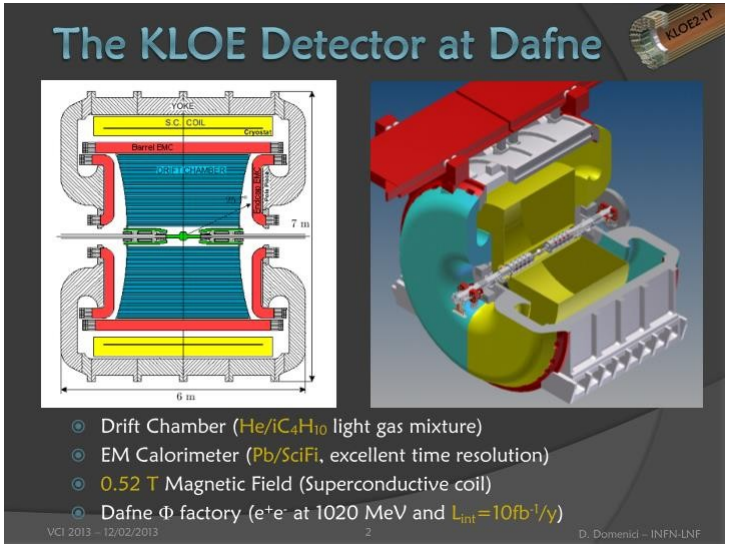
- Buffers are not the <final solution> they can overflow due to:
  - bursts
  - unusual event sizes
- Discard
  - local, or
  - "backpressure", tells lower levels to discard

Who controls the flow?

The FE (*push*) or the EB (*pull*)

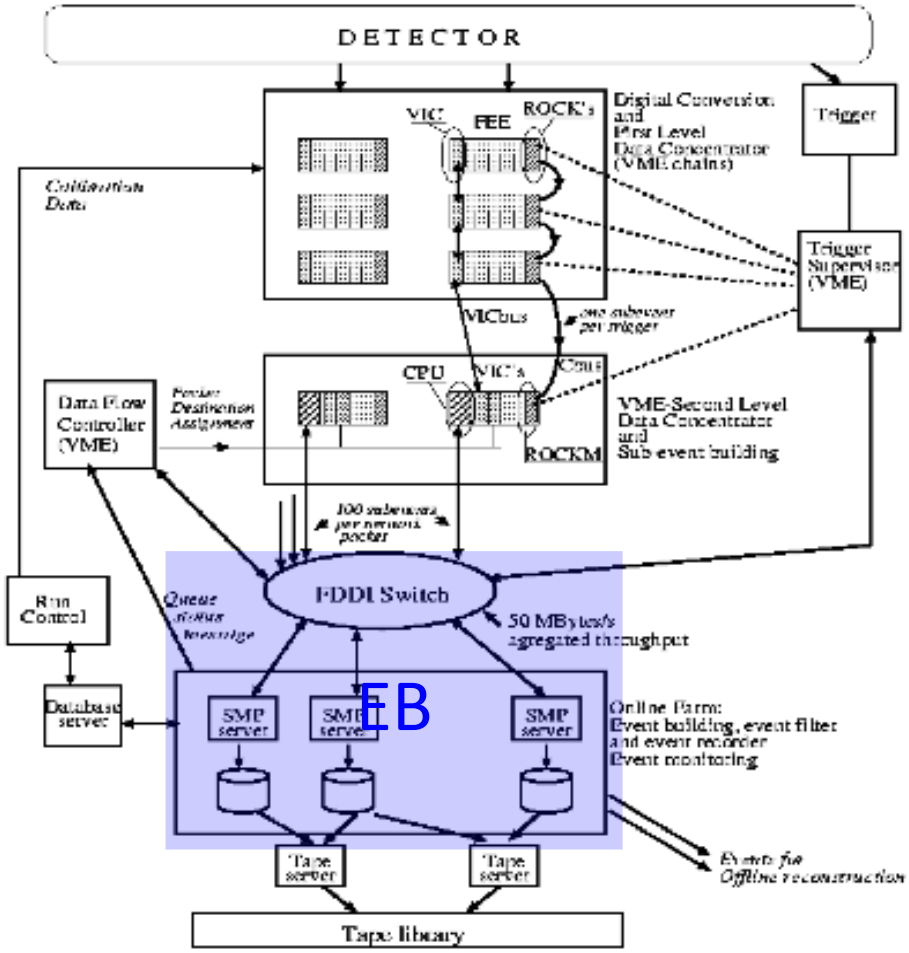
a *push* example: KLOE

- DAΦNE e^+e^- collider in Frascati
- CP violation parameters in the Kaon system
- "factory": rare events in a high-rate beam



- 10^5 channels
- 2.7ns crossing rate
  - rarely event overlap
  - "double hit" rejection
- high rate of small events
- L1 ~10^4 Hz
  - 2μs fixed dead time
- HLT ~10^4 Hz
  - ~COTS, cosmic rejection only
- 5kB/ev → 50MB/s [design]

KLOE



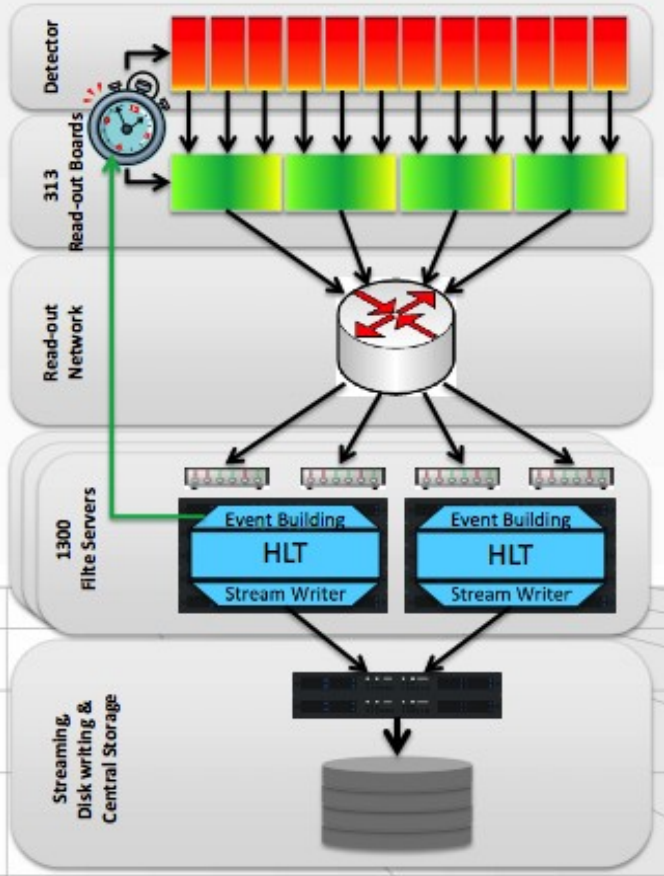
- deterministic FDDI network
- not real need for buffering at FE
- *push* architecture vs pull used in ATLAS
  see DAQ Software lecture
- try EB load redistribution before resorting to backpressure

Which LHC experiment has a somewhat similar dataflow architecture ?

LHCb: dataflow is network

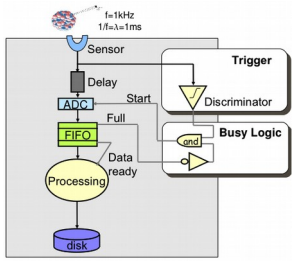
From Front-End to Hard Disk

- O(10^6) Front-end channels
- 300 Read-out Boards with 4 x 1 Gbit/s network links
- 1 Gbit/s based Read-out network
- 1500 Farm PCs
- >5000 UTP Cat 6 links
- 1 MHz read-out rate
- Data is pushed to the Event Building layer. There is no re-send in case of loss
- Credit based load balancing and throttling

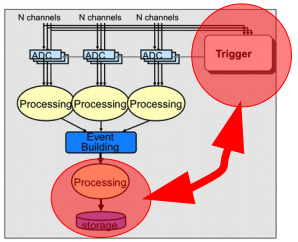


The LHCb Data Acquisition during LHC Run 1
CHEP 2013

more info in "TDAQ for the LHC experiments"



Trends



- Integrate synchronous, low latency in the front end
  - the limitations discussed do not disappear, but decouple (factorise)
  - all-HW implementation
  - isolated in replaceable(?) components
- Use networks as soon as possible

- Deal with dataflow instead of latency
- Use COTS network and processing
- Use "network" design already at small scale
  - easily get high performance with commercial components

Back to basics ?

- *(12) In [protocol] design, perfection has been reached not when there is nothing left to add, but when there is nothing left to take away.*

  RFC 1925 The Twelve [Networking] Truths

After adding all these levels of buffering, indirection, preselection, pre-preselection ...

... what if we threw it all away?

Well, sometimes we can, sometimes we can't.

see TDAQ for the LHC experiments

take care #1, lot of issues not covered:

- Hw configuration
- Sw configuration
- Hw control & recovery
- Sw control & recovery
- Monitoring
- …

take care #2:

- in average things (often) do work, but what about fluctuations/exceptions ?

Thank you for your patience …