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

ISOTDAQ 2017
Nikhef - Amsterdam Science Park
5 February 2017

- Next 5 slides are mute … you have just to read them (they are simple enough)

Roberto Ferrari
Istituto Nazionale di Fisica Nucleare

Roberto Ferrari
Istituto Nazionale di Fisica Nucleare

Oh my ! Yet another f…[1] Italian[2] !

(1) fabulous ... friendly ... fried … ?

(2) about 11.5+Kostas lectures (out of 27) covered by Italians

Roberto Ferrari
Istituto Nazionale di Fisica Nucleare

Oh my ! Yet another f…[1] Italian[2] !

(1) fabulous ... friendly ... fried … ?

(2) about 11.5+Kostas lectures (out of 27) covered by Italians

(students' homework)
discuss one of the following hypothesis:

baseline: what the probability of a statistical over-fluctuation ?

ultimate: what about "do the Italians do it better" ?

more seriously

more seriously

For some days,
unluckily [luckily, you may think] ,
I have been out of any voice.

Not clear I have got back some of it.

more seriously

For some days,
unluckily [luckily, you may think] ,
I have been out of any voice.

Not clear I have got back some of it.

Hard to be sure I will last till the end of the talk !

more seriously

For some days,
unluckily [luckily, you may think] ,
I have been out of any voice.

Not clear I have got back some of it.

Hard to be sure I will last till the end of the talk !

… as last resort ...

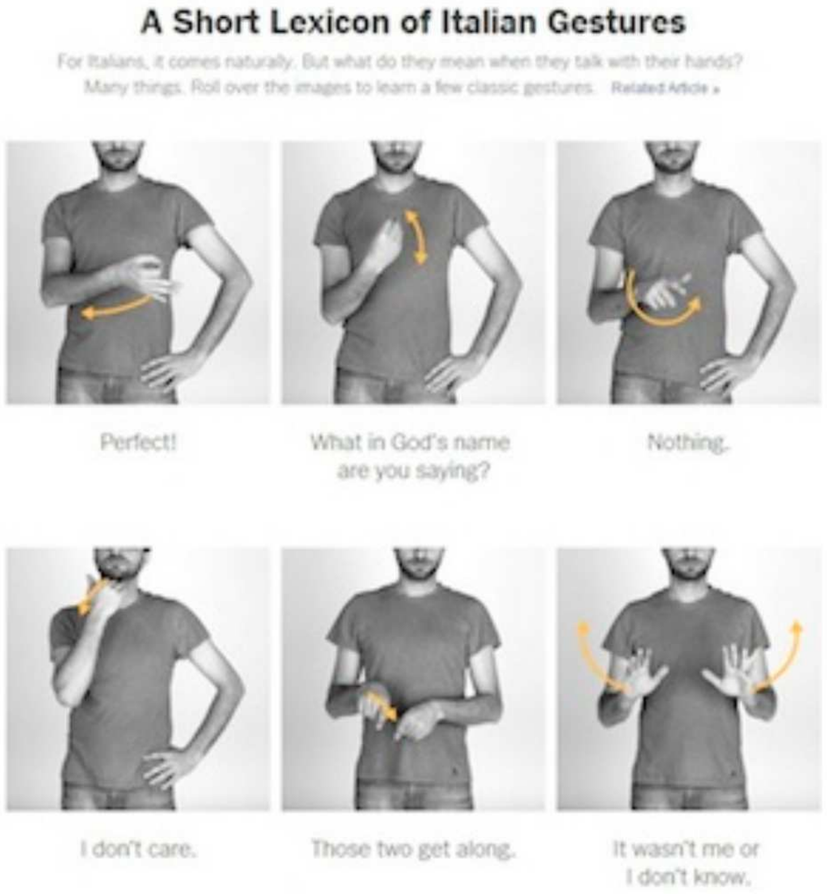
… I've thought you may be happy to practice the Italians' way …

… I've thought you may be happy to practice the Italians' way …

don't be scared … plenty of dictionaries on the web:

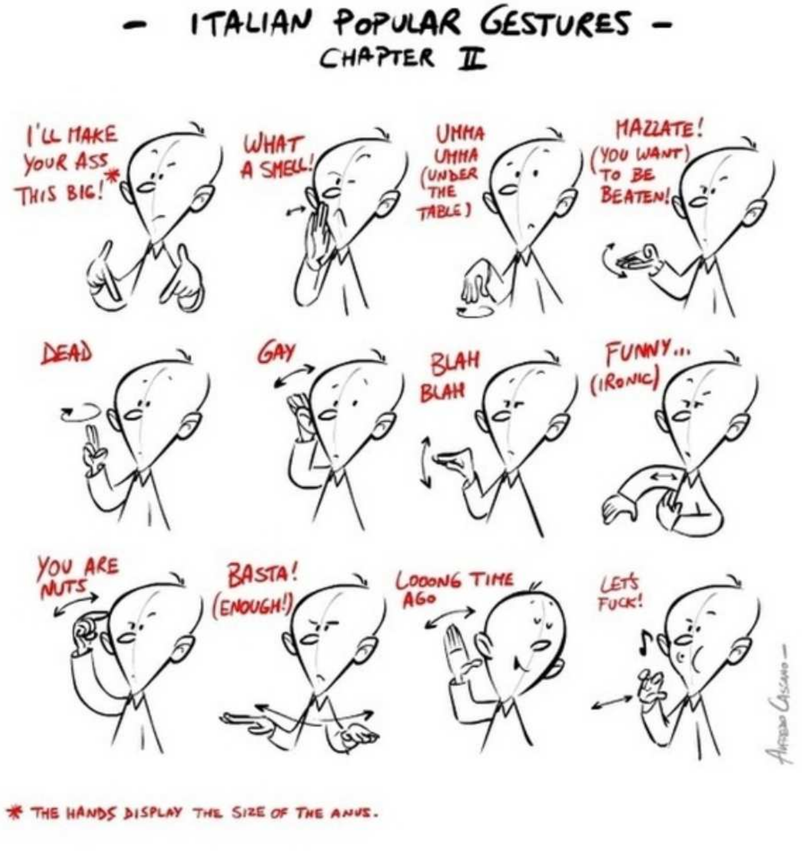
… I've thought you may be happy to practice the Italians' way …

don't be scared … plenty of dictionaries on the web:



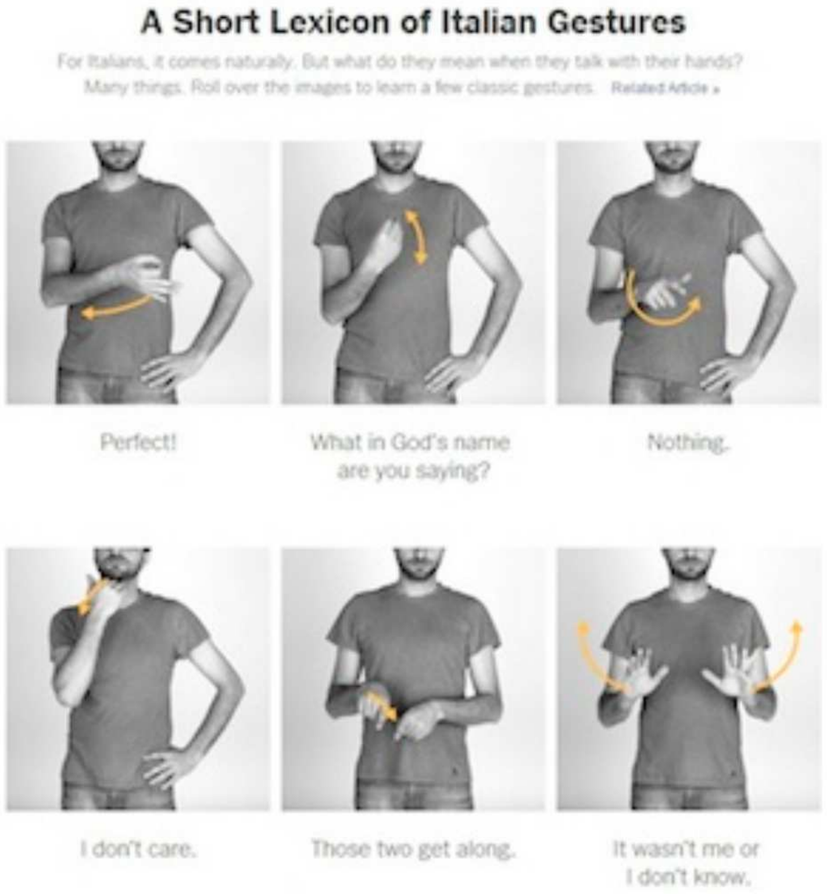
… I've thought you may be happy to practice the Italians' way …

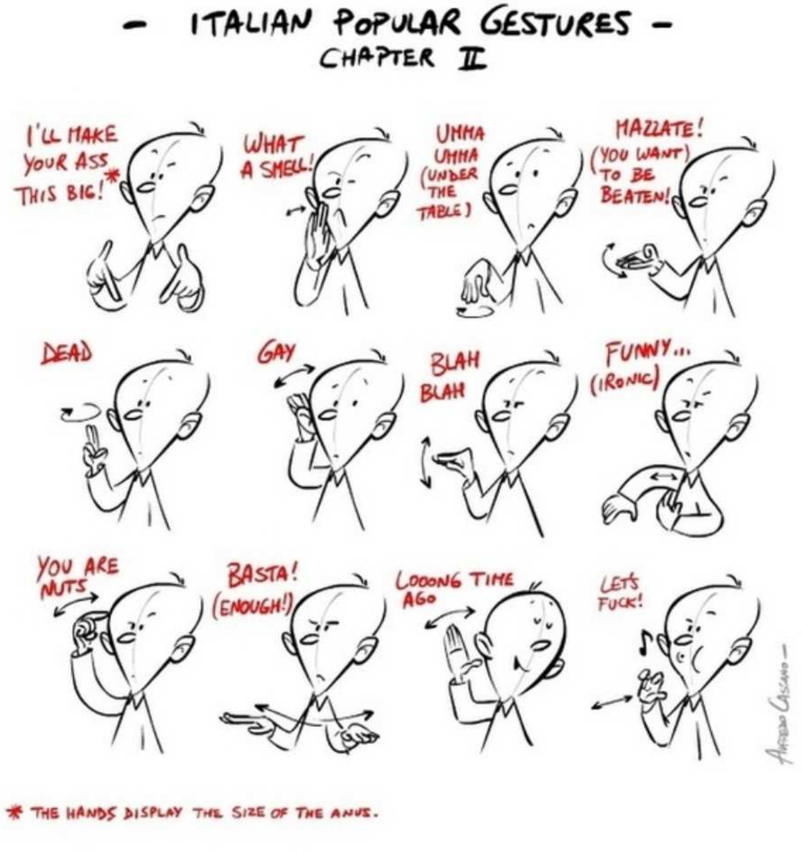
don't be scared … plenty of dictionaries on the web:



… I've thought you may be happy to practice the Italians' way …

don't be scared … plenty of dictionaries on the web:





(oops … be careful with some of them when doing exercises)

++more ++seriously

++*more* ++*seriously*

(1)

++*more* ++*seriously*

(1)

→ hope to give you something sensible ←

++*more* ++*seriously*

(1)

→ hope to give you something sensible ←

(2)

++*more* ++*seriously*

(1)

→ hope to give you something sensible ←

(2)

→ but, please, don't take anything at face value ←

++more ++seriously

(1)

→ hope to give you something sensible ←

(2)

→ but, please, don't take anything at face value ←

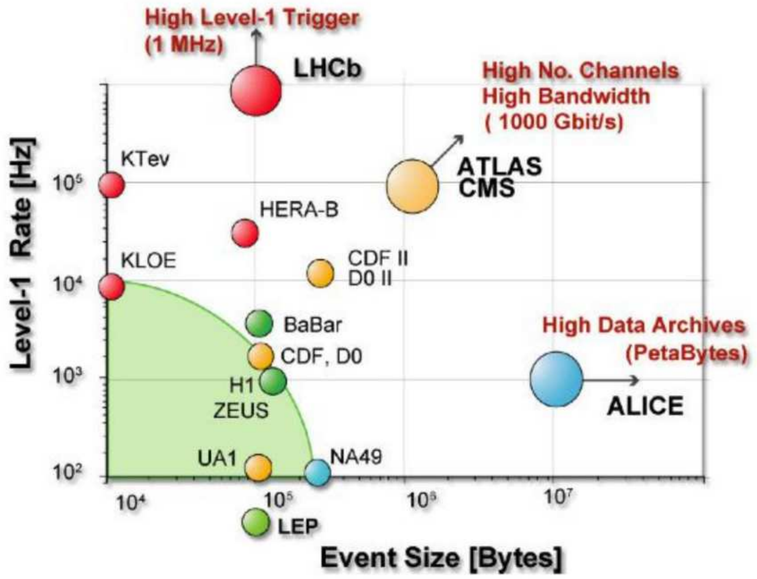
just aiming at enlightening some critical issues

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

UNIVERSITY OF JOHANNESBURG

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

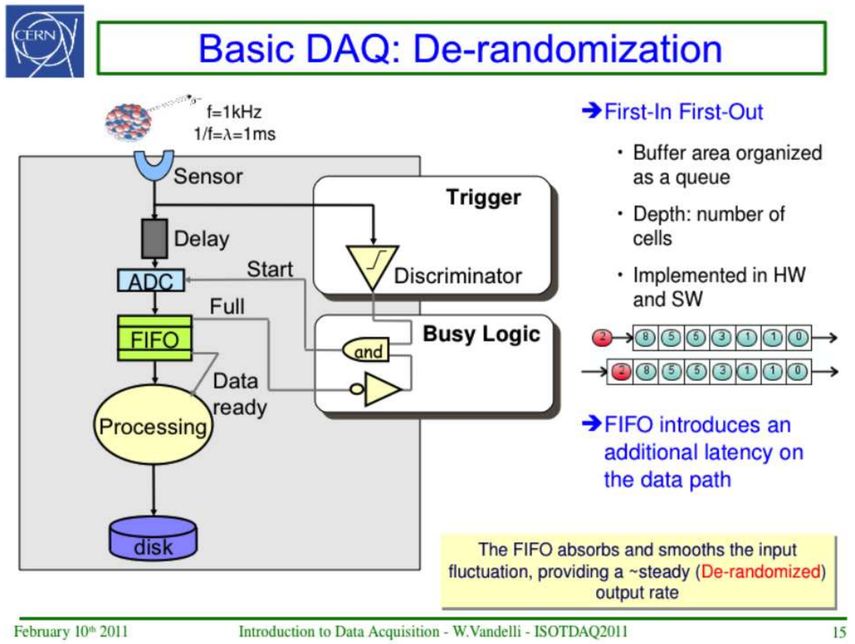
CERN



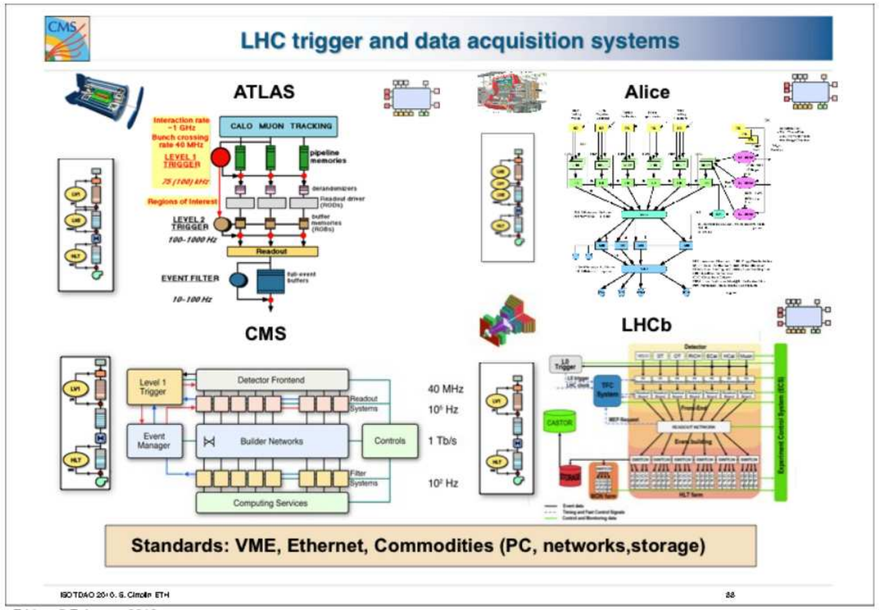
ISOTDAQ2015, Rio de Janeiro TDAQ Scaling - Sergio Ballestrero Page 1

Trying to move …

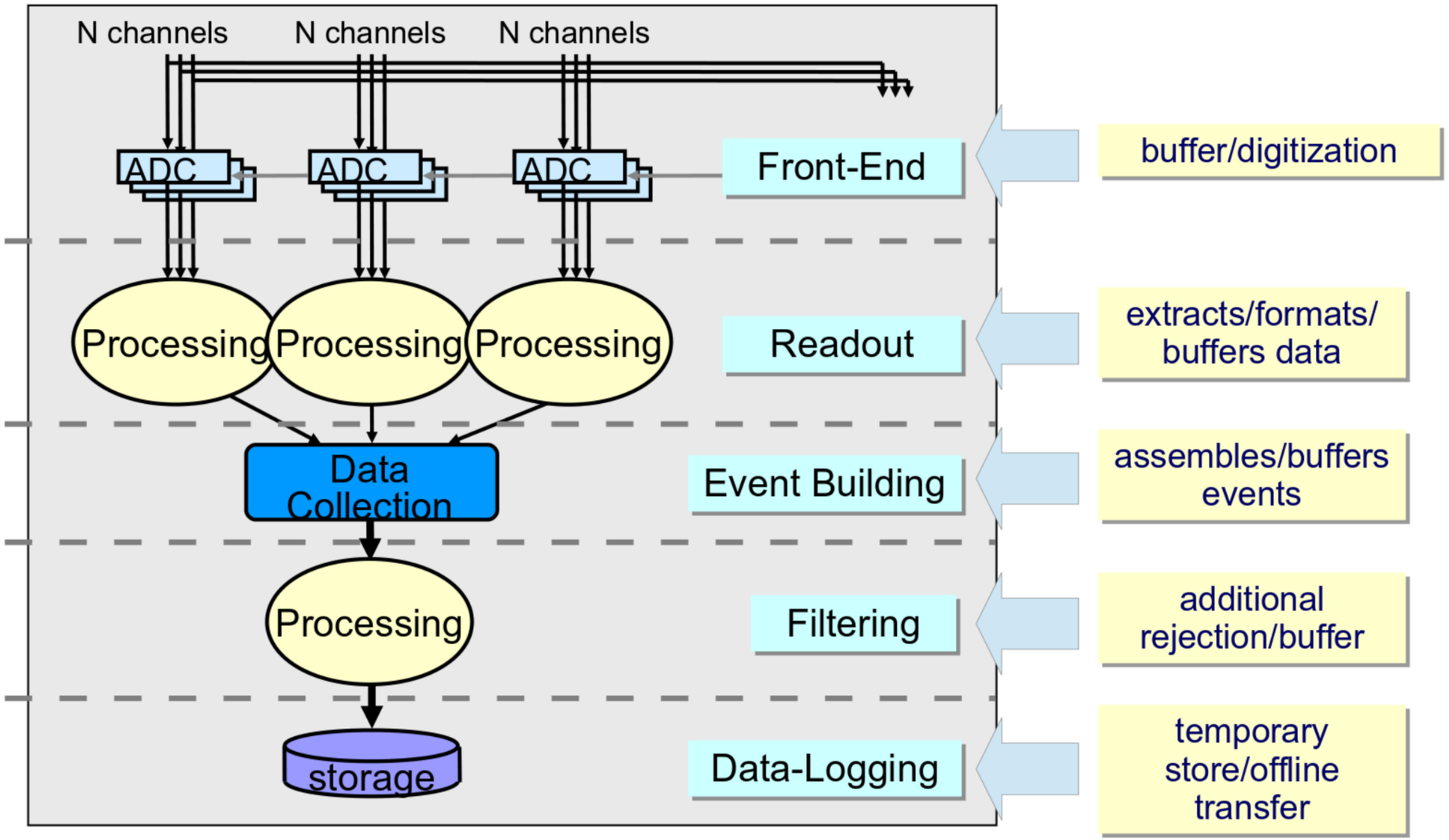
from here:



to here:



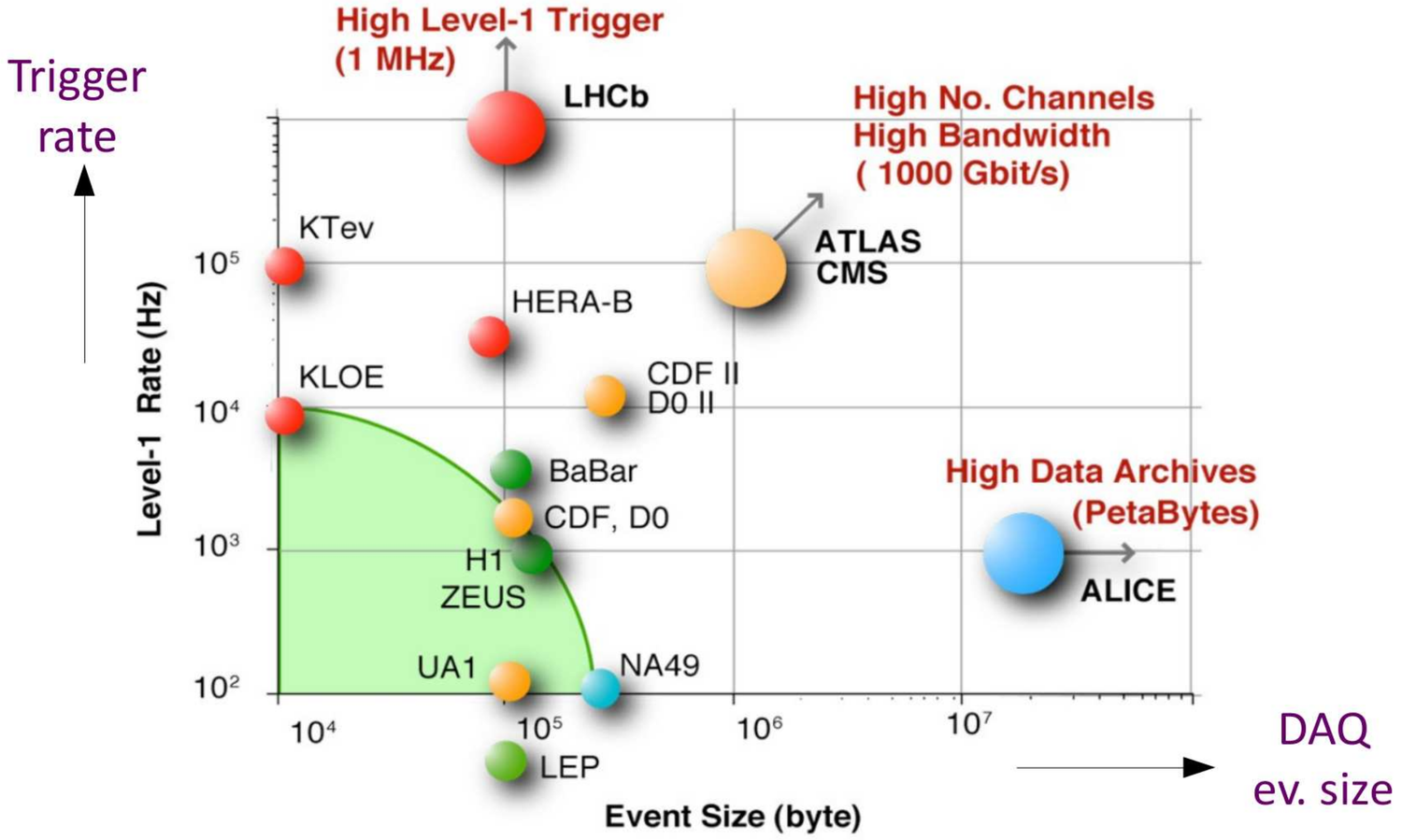
Medium/Large DAQ: constituents



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

Trigger & DAQ in HEP



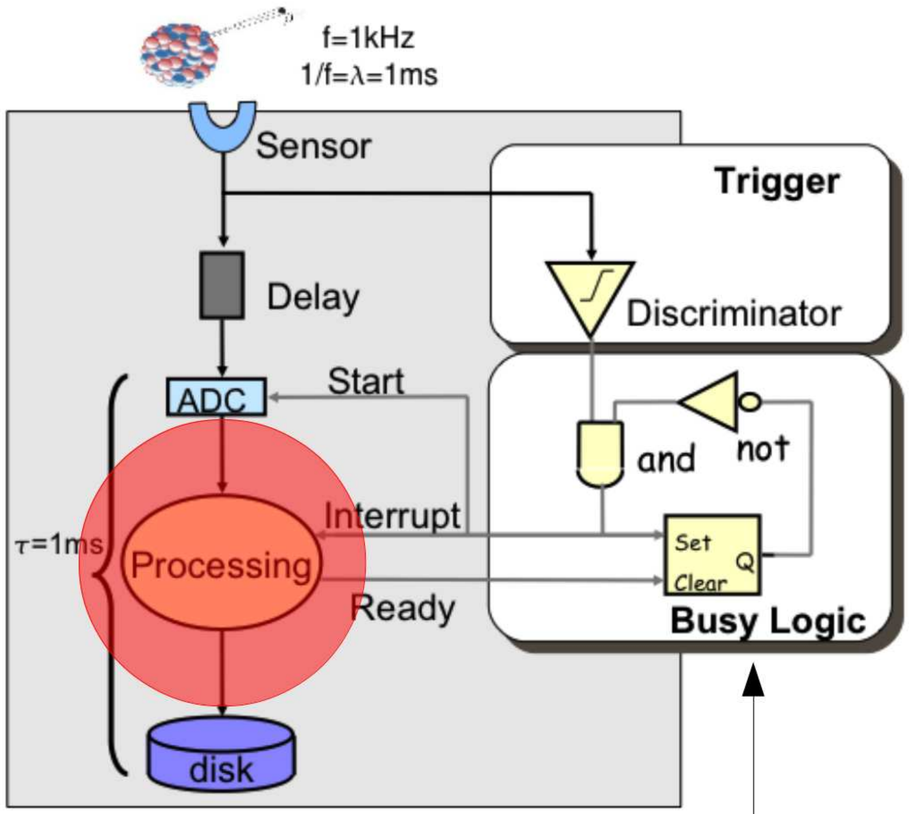
different issues → different solutions

no magic, unique, solution for all cases

step one: increase rate

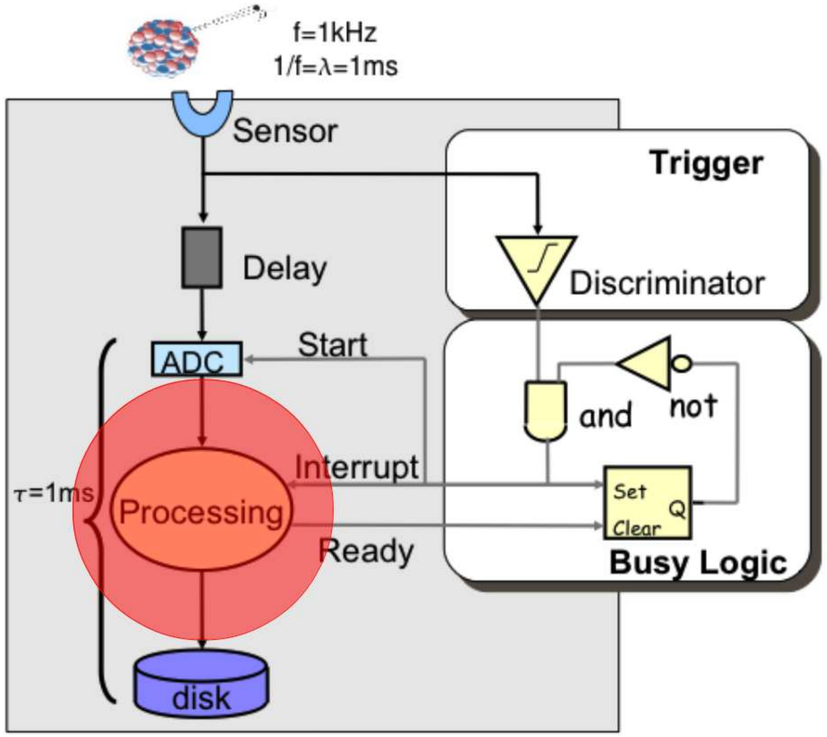
Single-event readout:

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



Dead Time → de-randomise

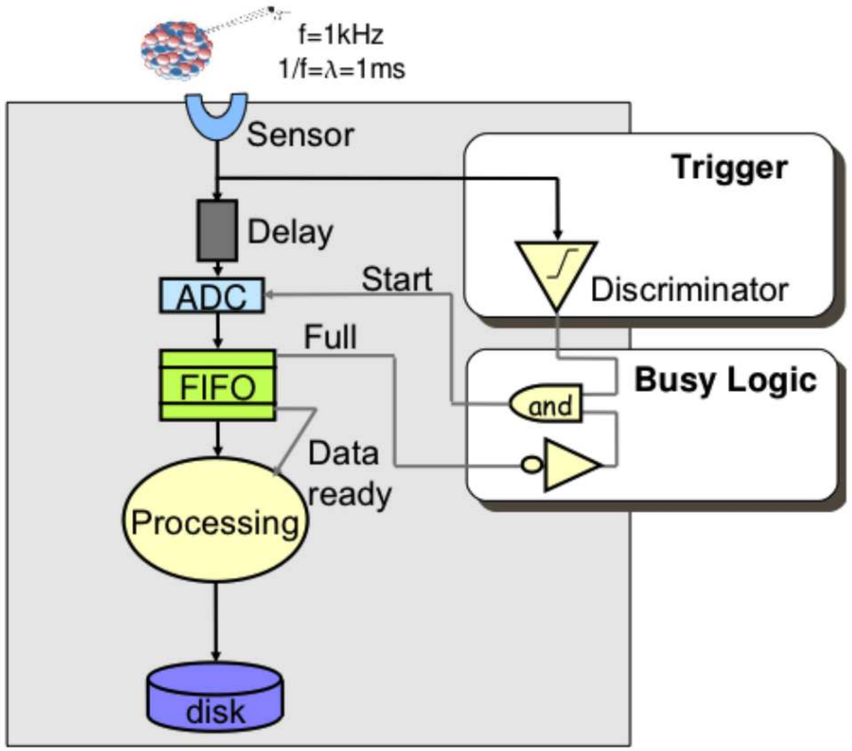
- Processing → bottleneck



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

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

- Buffering allows to decouple problems



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

[N = buffer depth]

[r = f(out)/f(in)]

LHC (collider) → synchronous

… nevertheless, high luminosity & high cross sections

→ high rate, high-pileup, large events:

➔ most events uninteresting

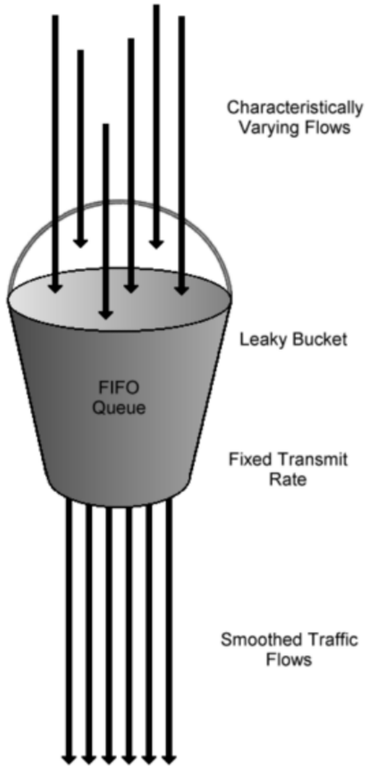
➔ good events (triggers) arrive uncorrelated (unpredictable)

➔ de-randomisation is still needed

➔ dataflow is an issue

does buffering solve all problems ?

- FIFO
  - filled with very variable input flow
  - emptied at smoothed output flow
- → the Leaky-Bucket problem
-
- Q: how often may overflow ?



off-topic: some very basic queueing theory

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

P_k : % time with k events in ; P_N = no space available → dead time

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

rate(j→j+1) = $\lambda \cdot P_j$ (fill at rate λ)

rate(j+1→j) = $\mu \cdot P_{j+1}$ (empty at rate μ > λ)

steady state: $\lambda \cdot P_j = \mu \cdot P_{j+1} \Rightarrow P_{j+1} = \rho \cdot P_j = \rho^{j+1} \cdot P_0$ where $\rho = (\lambda/\mu) <\sim 1$

for $\rho \sim 1 \Rightarrow P_j \sim P_{j+1} \Rightarrow \sum P_k \sim (N+1) \cdot P_0 = 1 \Rightarrow P_0 \sim P_N \sim 1/(N+1)$

$\Rightarrow$ dead time $\sim 1/(N+1)$

want ≤ 1% ⇒ N ≥ 100

off-topic: some very basic queueing theory

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

P_k : % time with k events in ; P_N = no space available → dead time

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

rate(j→j+1) = $\lambda \cdot P_j$ (fill at rate λ)

rate(j+1→j) = $\mu \cdot P_{j+1}$ (empty … μ > λ)

steady state: … $_{j+1}$ ⇒ $P_{j+1} = \rho \cdot P_j = \rho^{j+1} \cdot P_0$ where $\rho = (\lambda/\mu) <\sim 1$

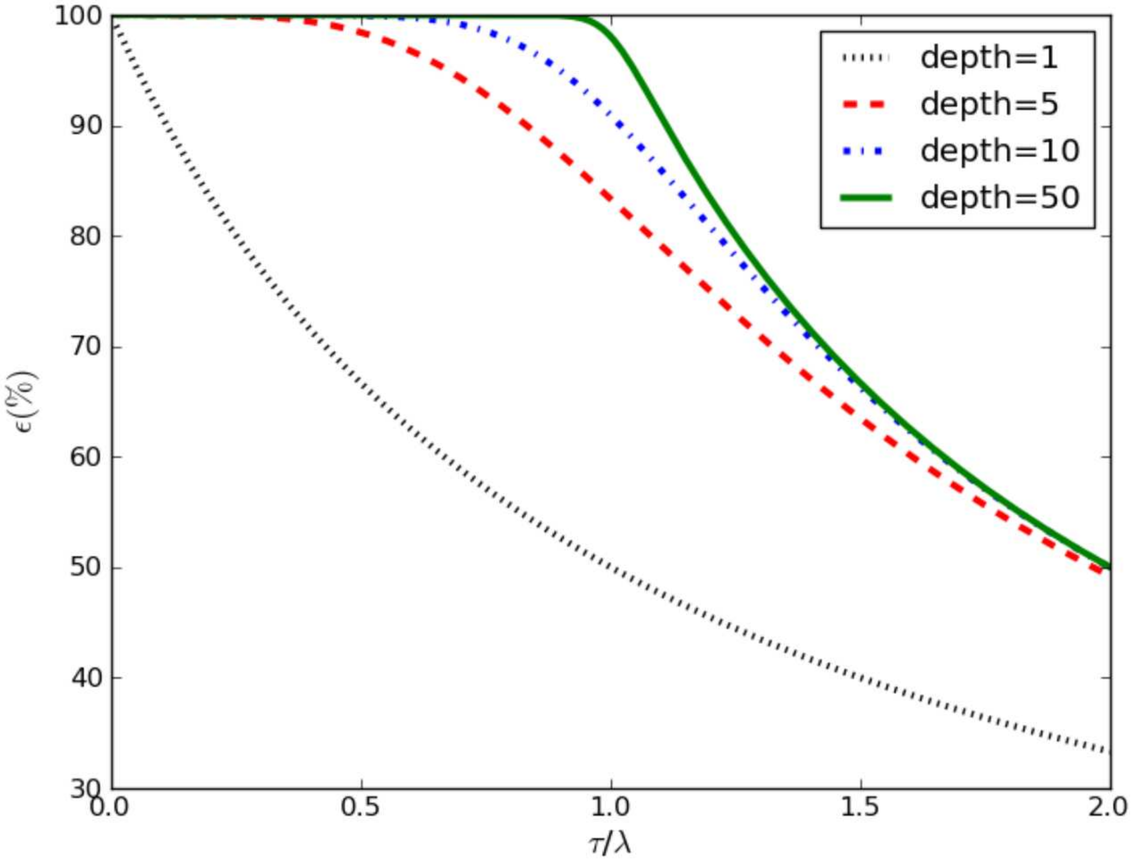
… 1 ⇒ $P_j \sim P_{j+1}$ ⇒ $\sum P_k \sim (N+1) \cdot P_0 = 1$ ⇒ $P_0 \sim P_N \sim 1/(N+1)$

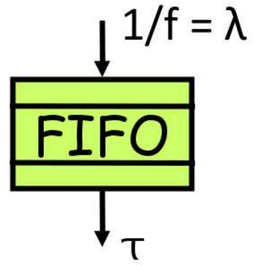
⇒ dead time ~ 1/(N+1)

want ≤ 1% ⇒ N ≥ 100

Take care: analytic calculation possible for pretty simple systems only

de-randomisation





- We can now attain a FIFO efficiency ~100% with:
  - $\tau \sim 1/f$
  - "moderate" buffer size
- Two degrees of freedom to play with
- This dead time often managed by trigger system itself ("complex dead time")

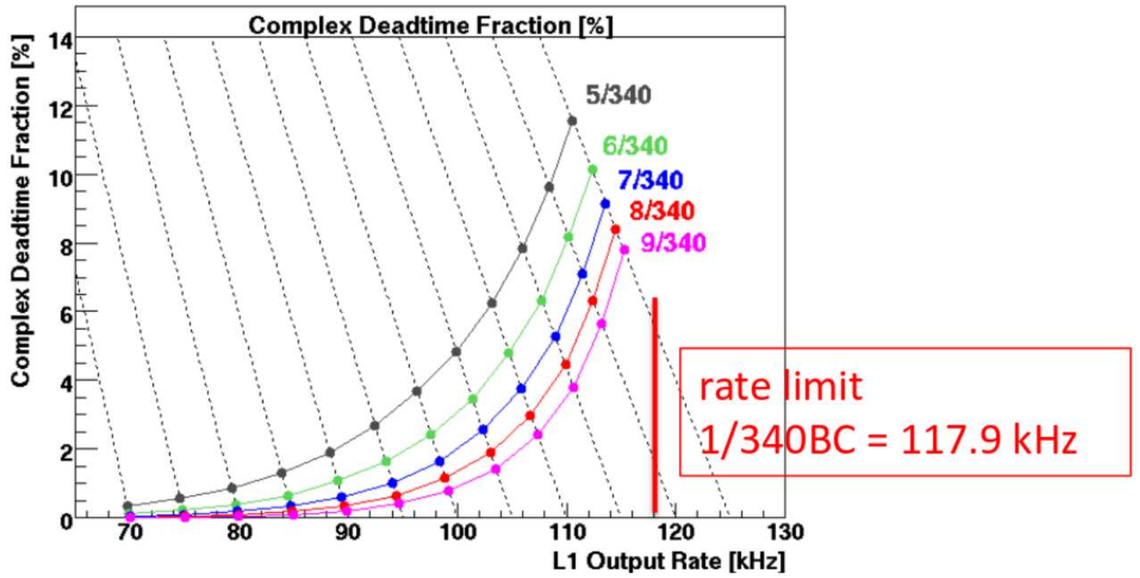
Complex Dead Time

1) Simple dead time: avoid overlapping (conflicting) readout window

2) Complex dead time: avoid overflow in front-end buffers (protection against trigger bursts)

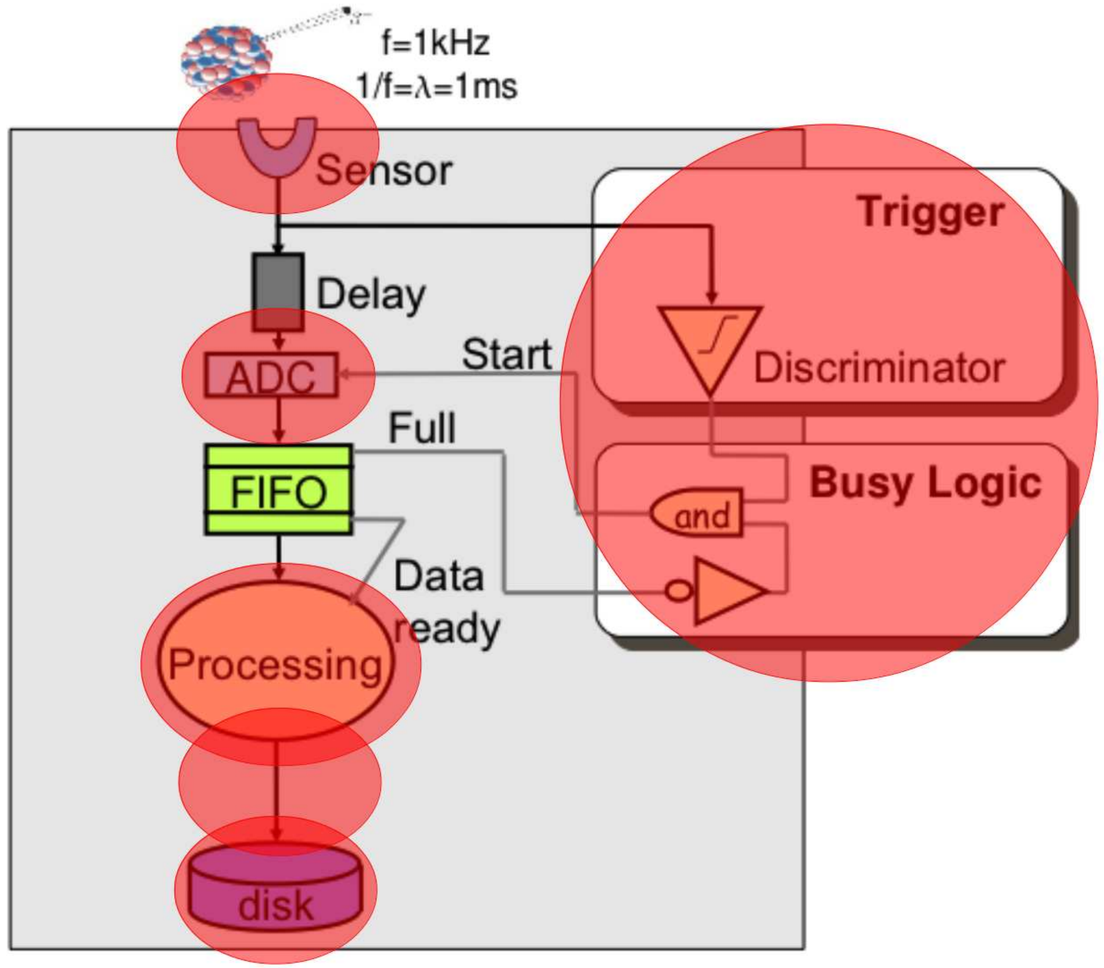
e.g. ATLAS uses simple leaky-bucket algorithms with 2 parameters:

max X triggers (X = FIFO depth) in any (sliding) time window = (X*readout time)



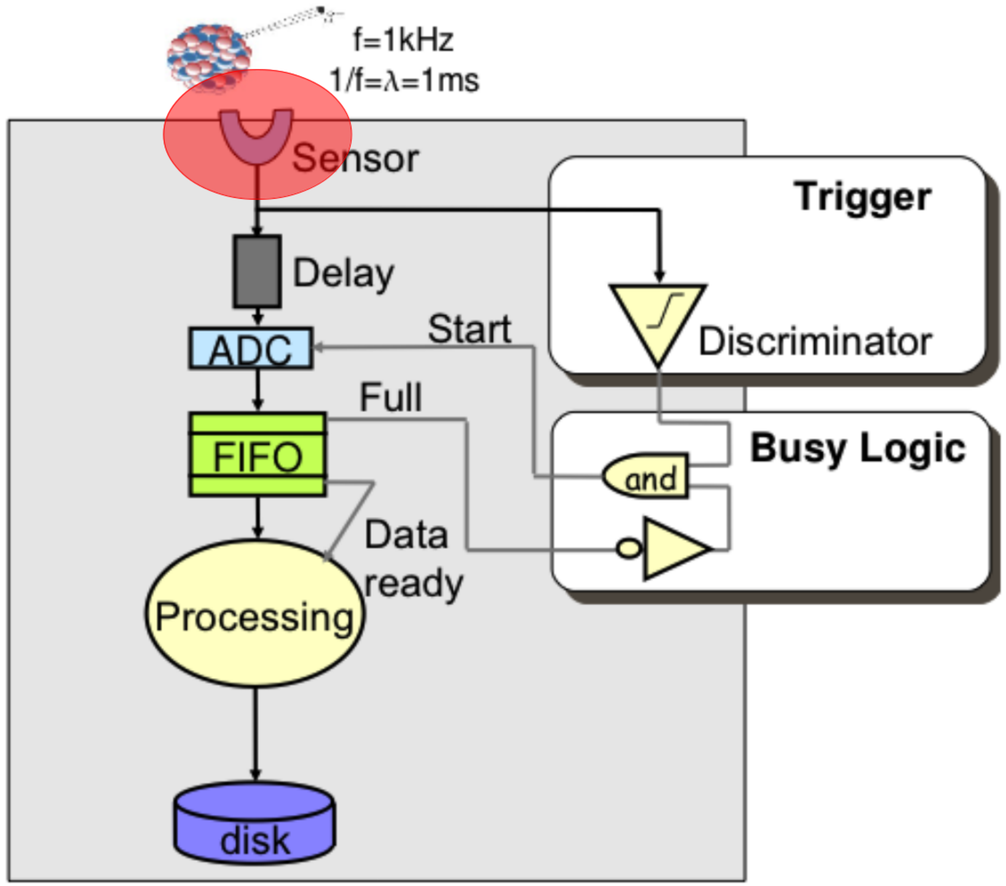
Game over ?

Even in a simple DAQ there are many other possible limits



→ the sensor

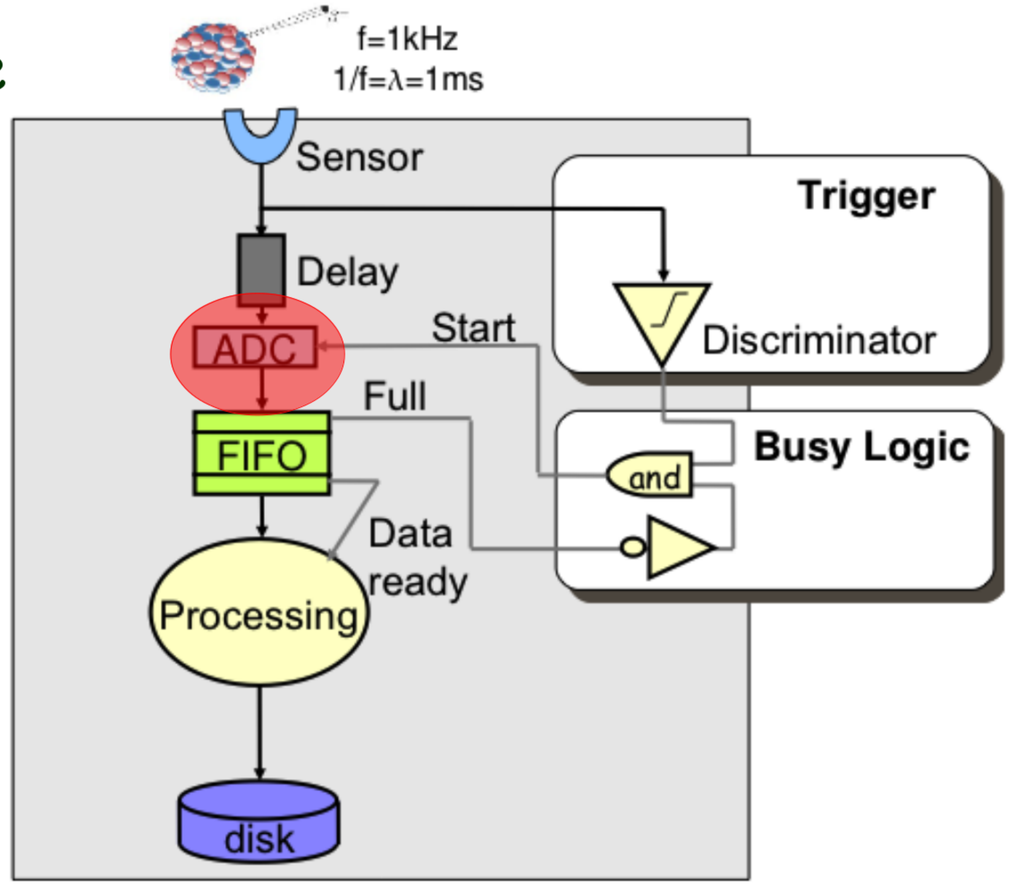
- Sensors are limited by physical processes, e.g.
  - drift times in gases
  - charge collection in Si
- (possibly) choose fast processes
- analog F.E. imposes limits as well
- split the sensors, each gets less rate: "increase granularity"



→ the ADC

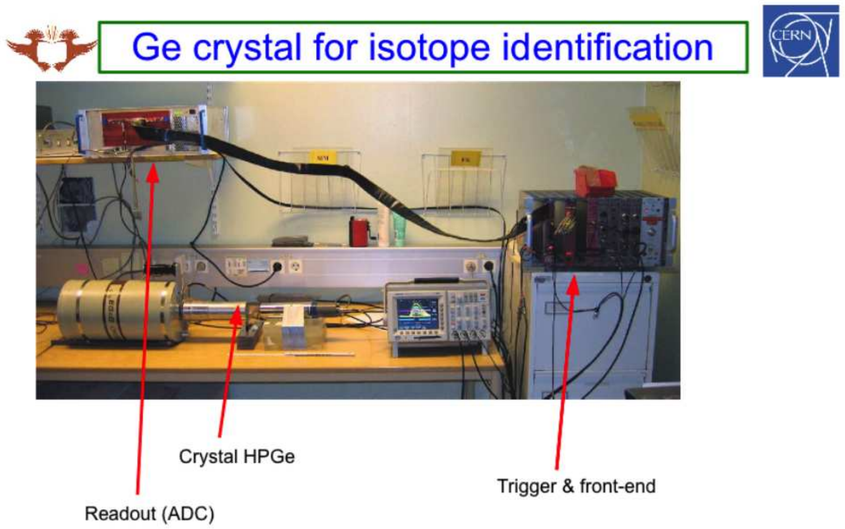
- A/D F.E. is also limited
- Faster ADCs pay the price in precision (# of bits) and power consumption
- Alternatives:
  - analog buffers
- You may need integration (or sampling) over quite some time

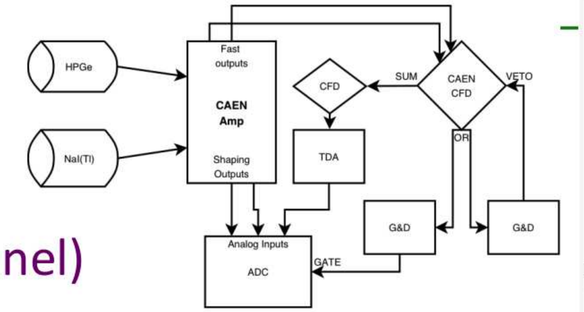
[see Detector Readout and FE lectures]

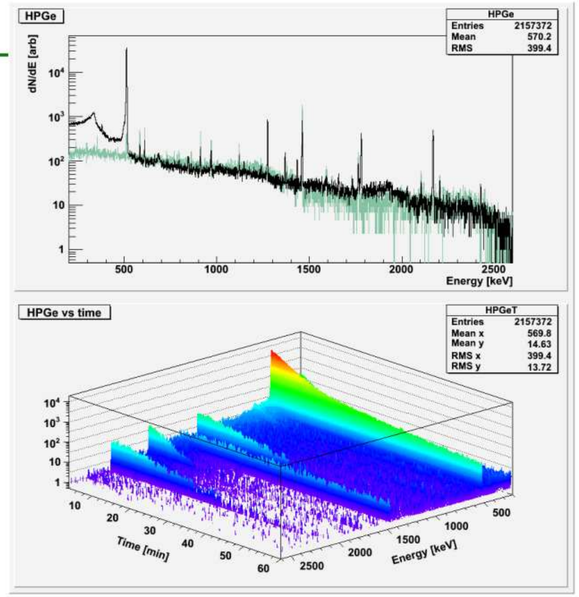


An example

- 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

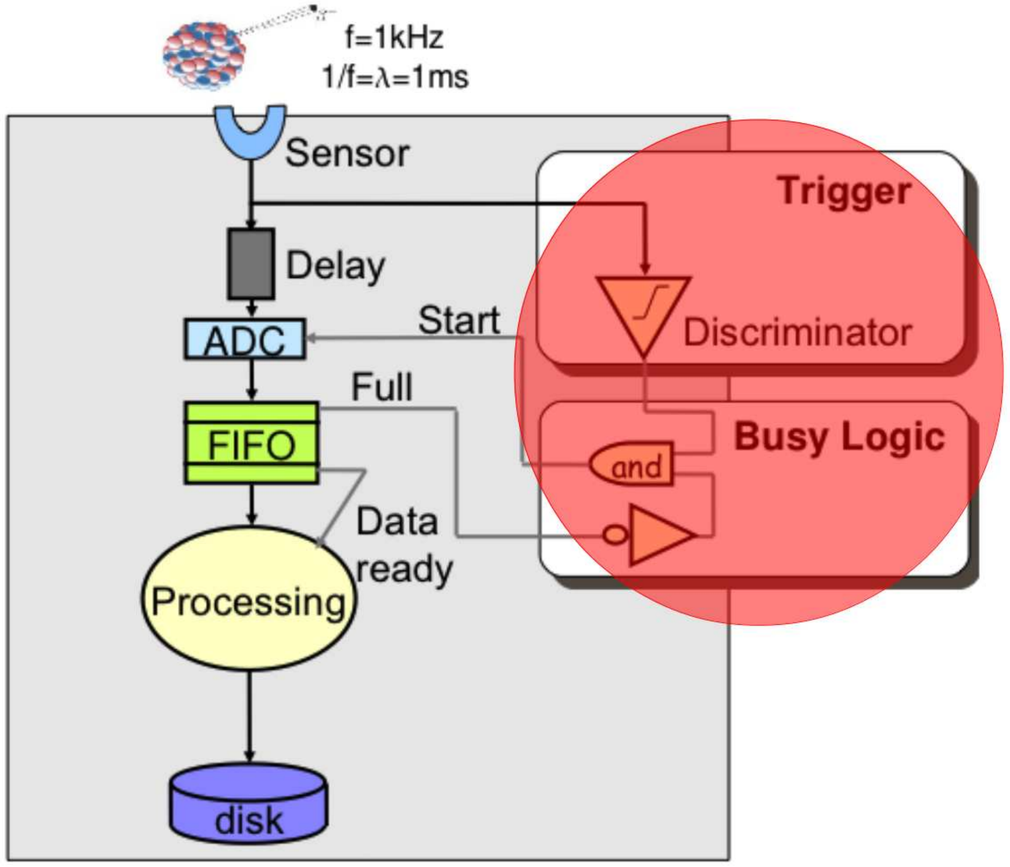






→ the trigger

- 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 ←

DREAM (2006→): a Testbeam Case

R&D on dual-readout calorimetry, setup:

- Crystals
- Scintillating/cherenkov fibers in lead/copper matrices
- Scintillator arrays as shower leakage counters
- Trigger/veto/muon counters
- Precision chamber hodoscope

… always evolving

Acquiring: waveforms, total charge, time information

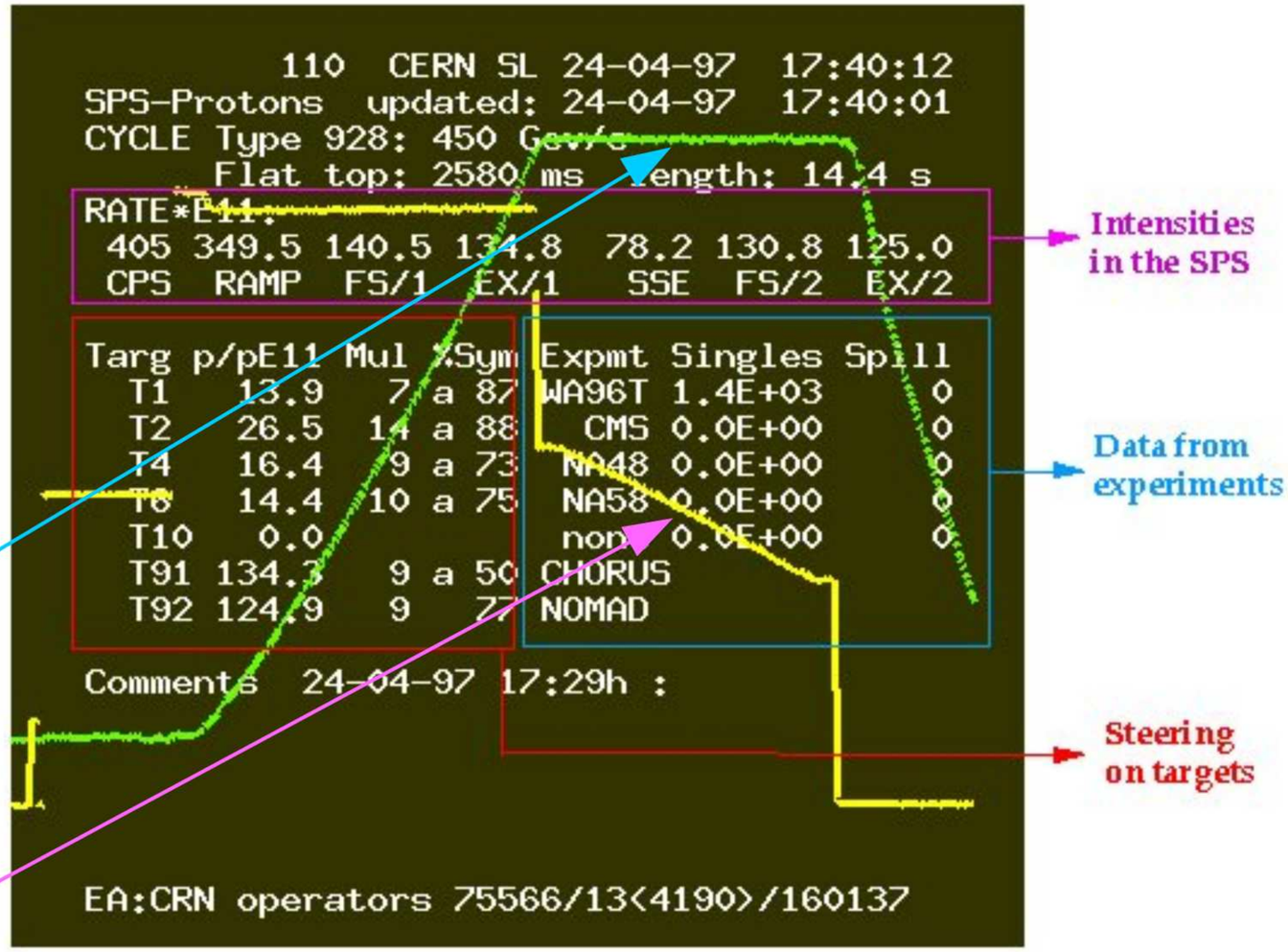
DREAM (2006→): a Testbeam Case

a possible SPS cycle

duty cycle:

~2 s / 14.4 s

(flat top)



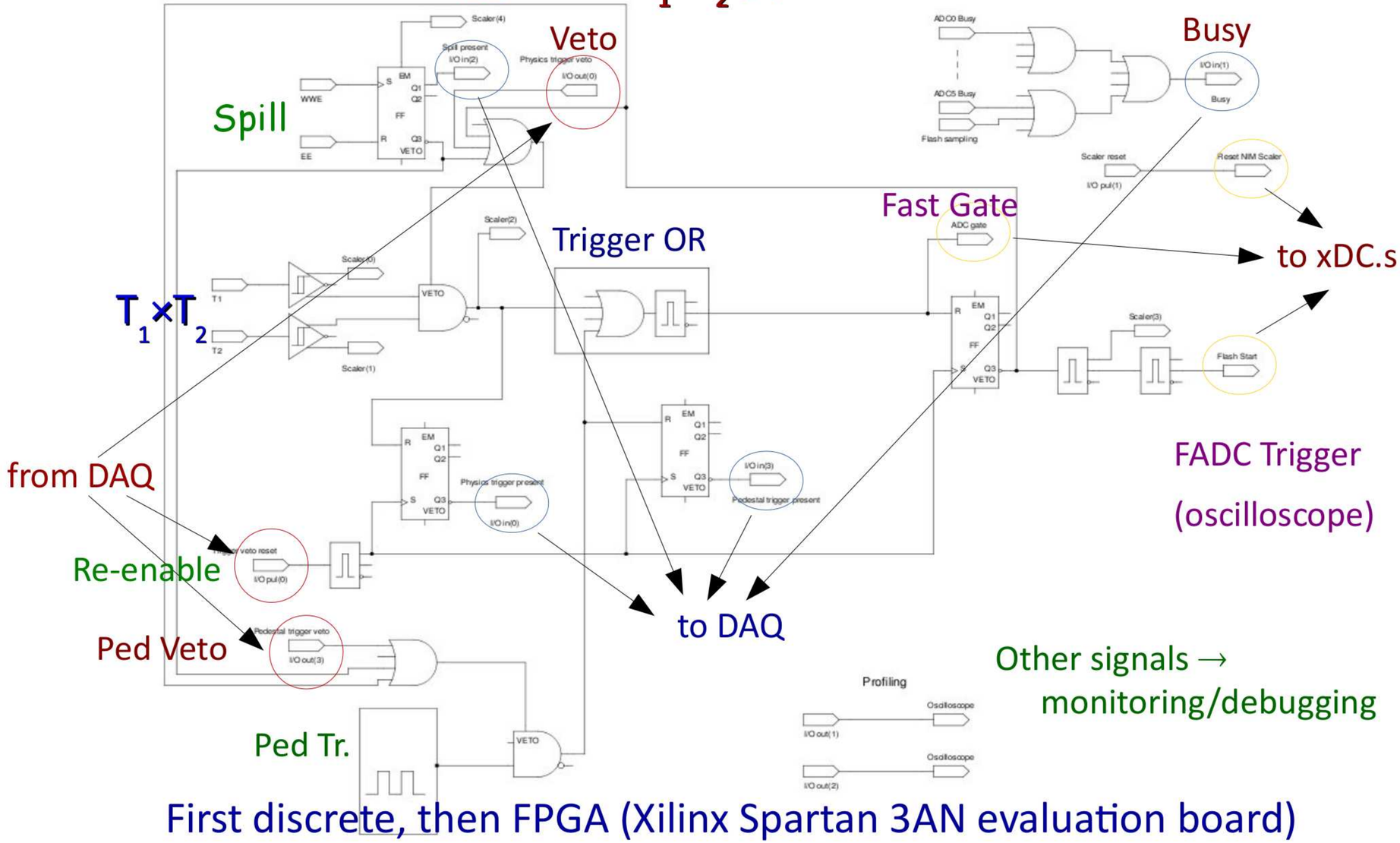
flat top

slow extraction

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

Spill-driven (asynchronous) Trigger

Trigger = $\bar{V} \times T_1 \times T_2$ | ped



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) 5 Gs/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", scientific linux desktop)

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

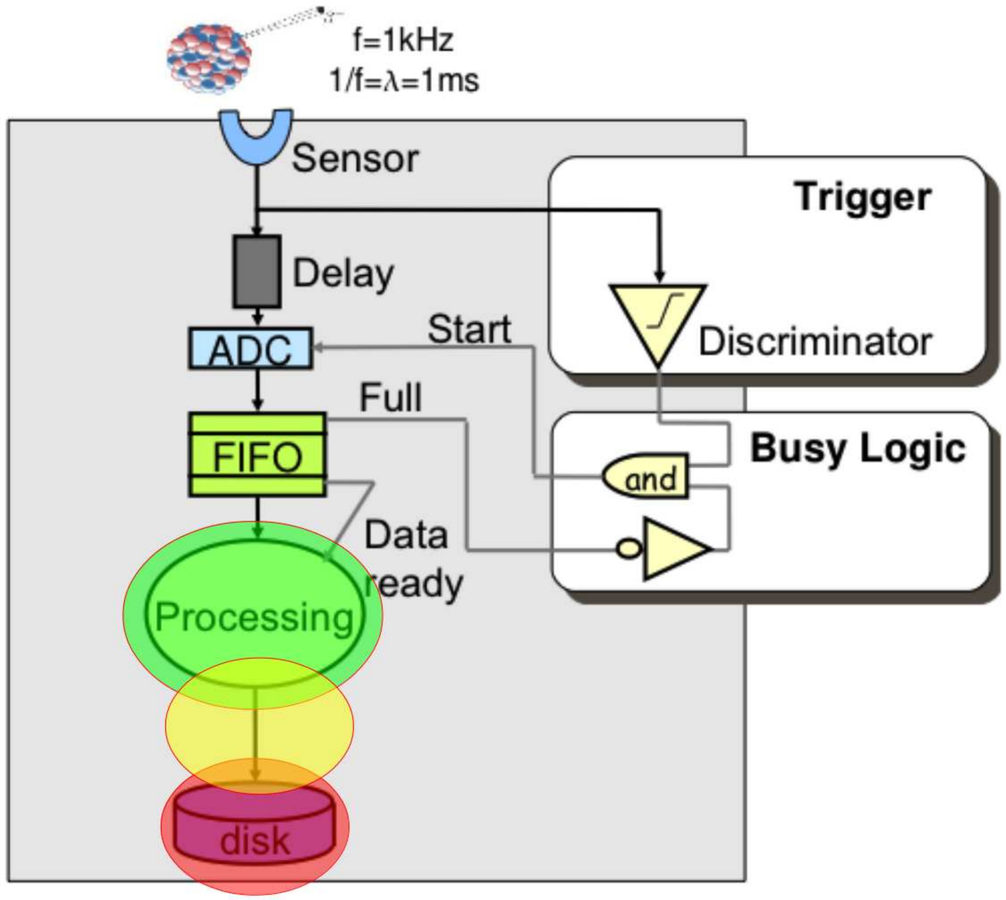
b.1) monitor data (produce root files)

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

rate ~ O(1 kHz)

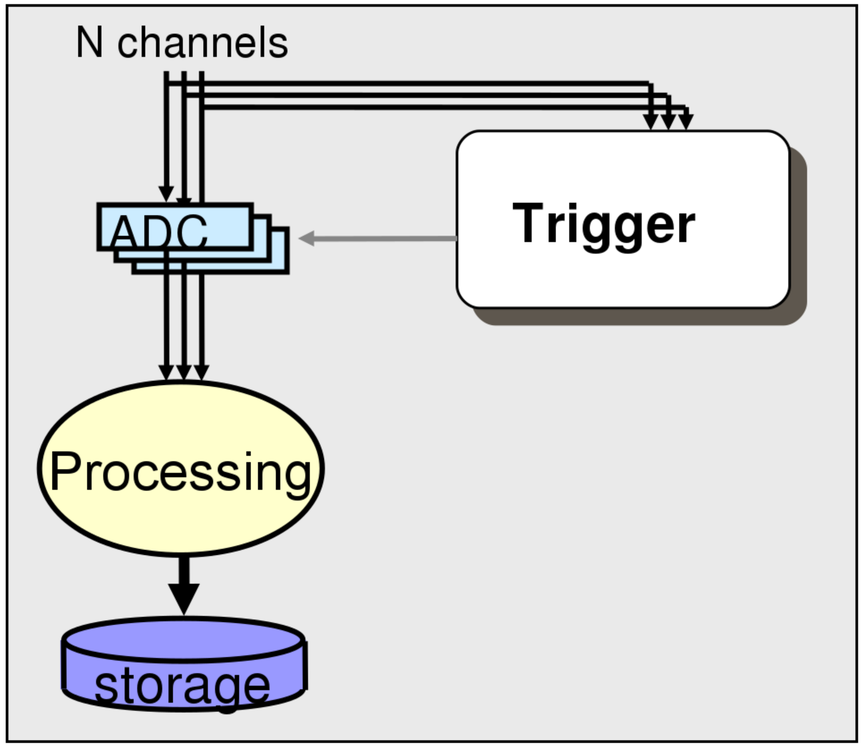
→ the dataflow

- 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

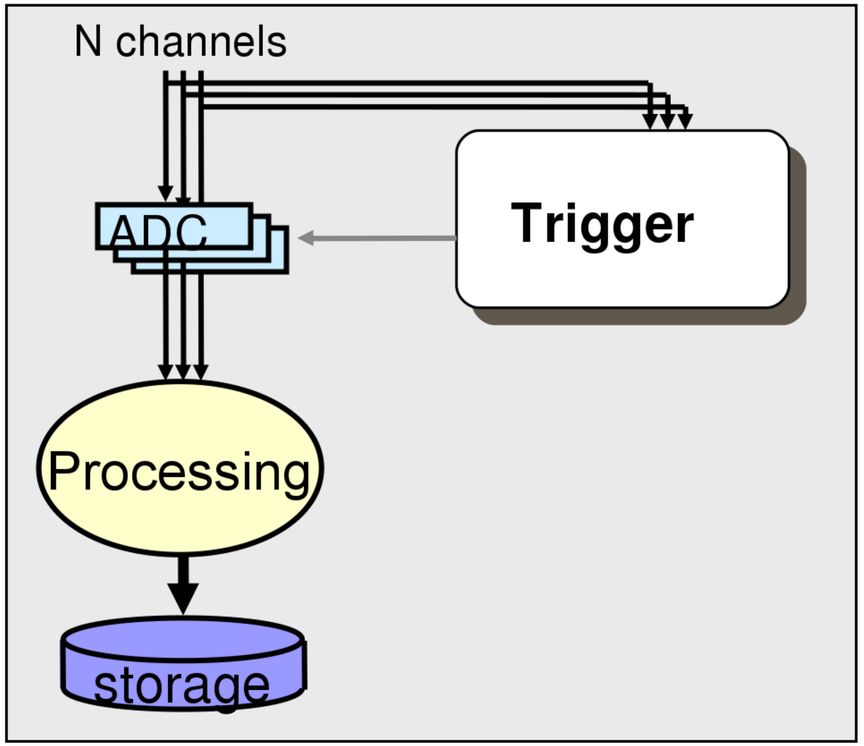


step two: increase # of sensors

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

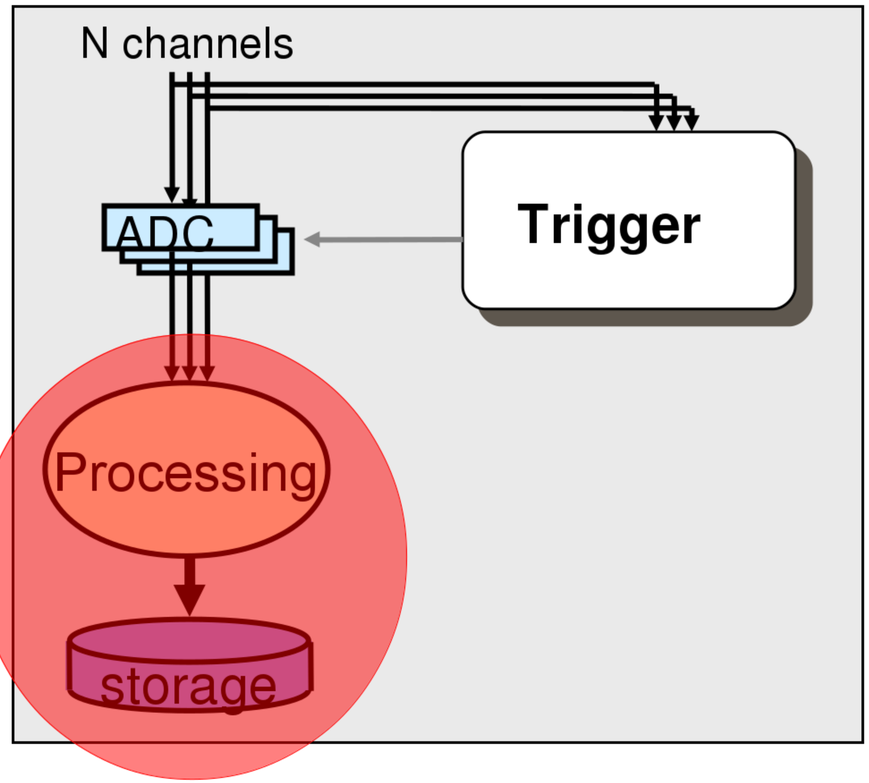


multi-channels, single PU



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

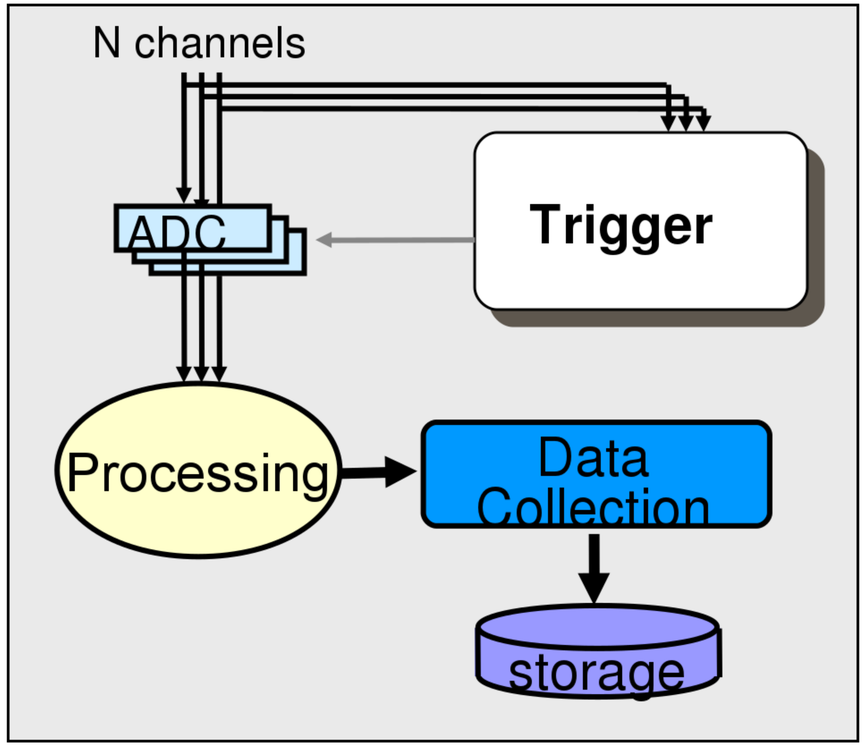
bottlenecks: PU and storage



- 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!

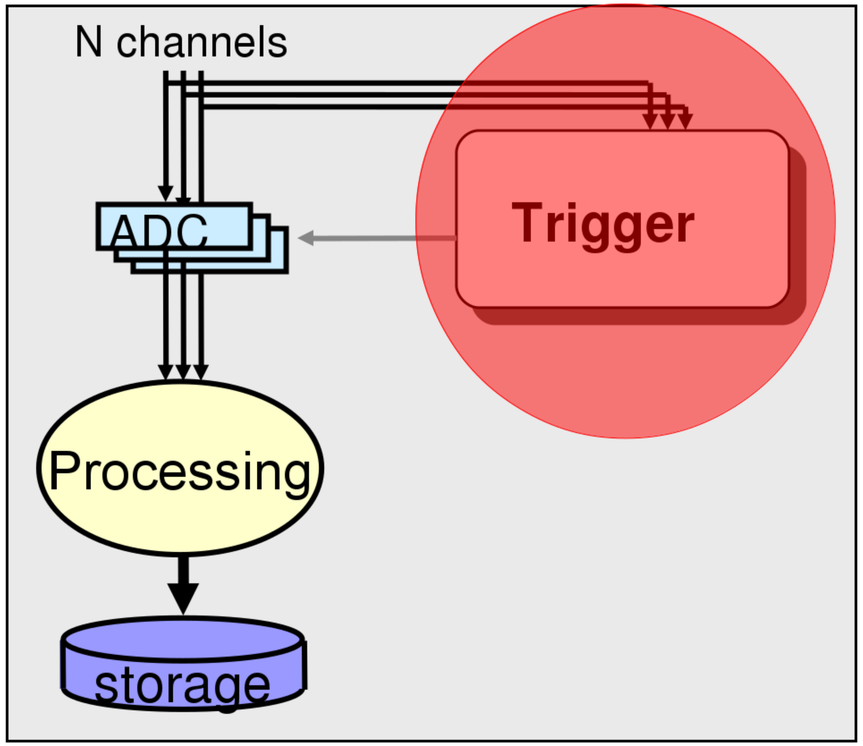
Laptop SATA disk: 54MB/s; USB2: ~30MB/s

→ 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

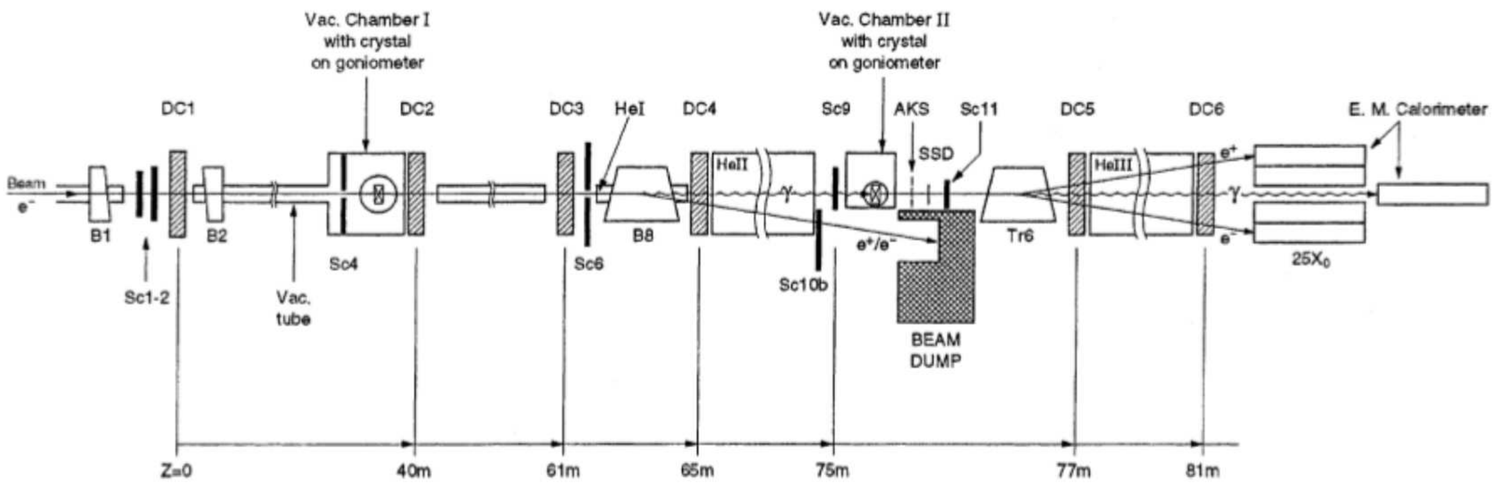
bottlenecks: trigger



- 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

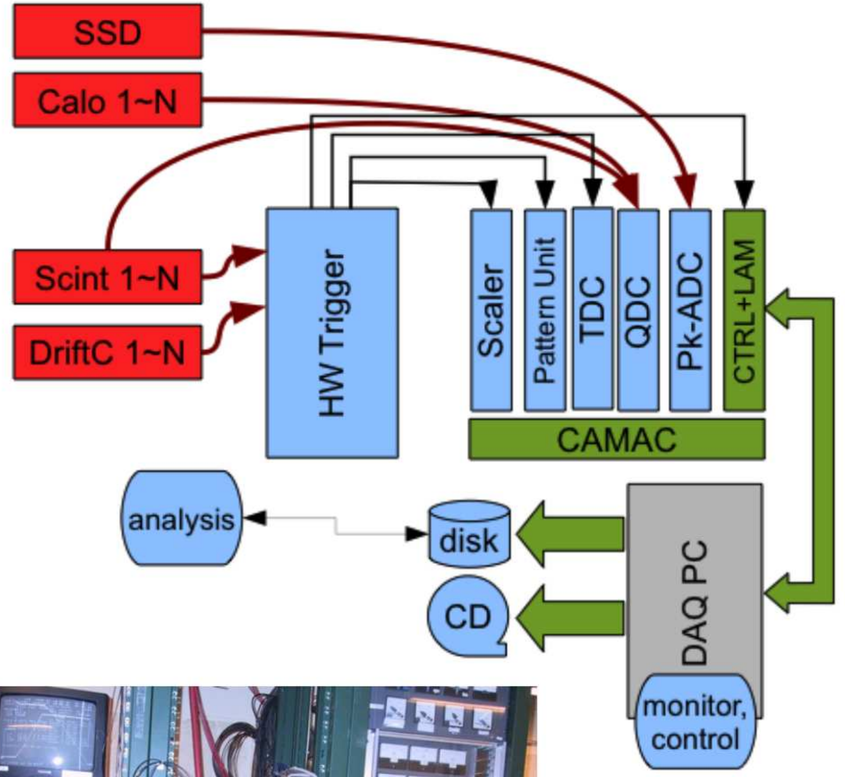


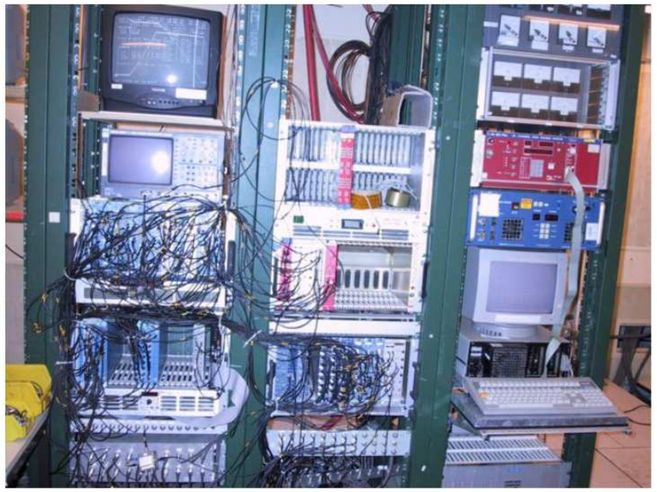
- Radiation processes: coherent emission in crystals and structured targets, LPM suppression...
- 80/120 GeV e- from CERN SPS slow extraction
- 2s spill every 13.5s

- 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

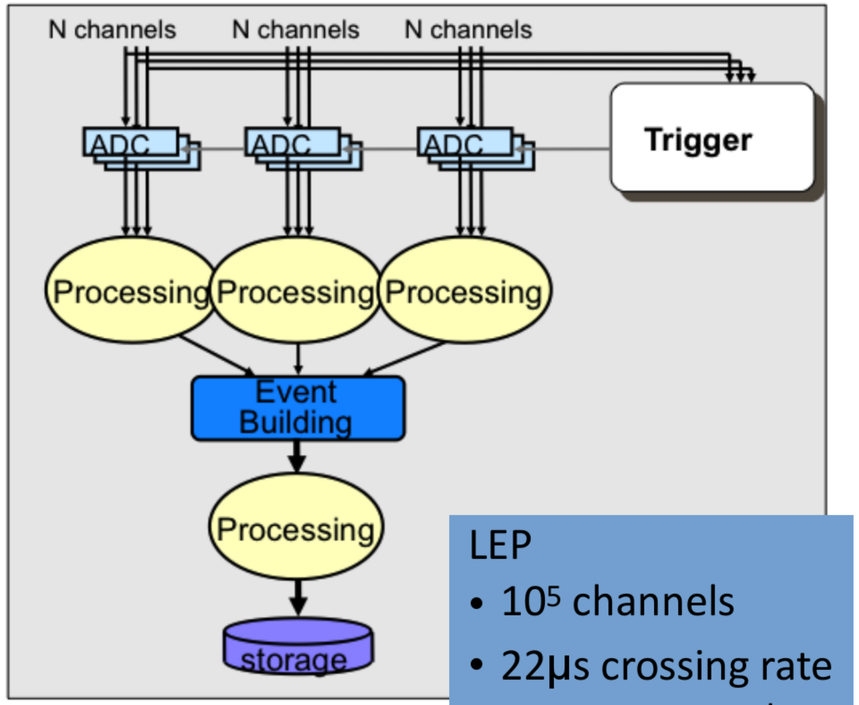
- 30-40 TDC, 6-16 QDC, 0-2 PADC
  (depending on measurement)
- CAMAC bus
  1 MB/s, no buffers, no Z.S.
- single PC readout
- NIM logic trigger
  (FPGA since 2009)
  - pileup rejection
  - fixed deadtime





step three: multiple PU (SBC)

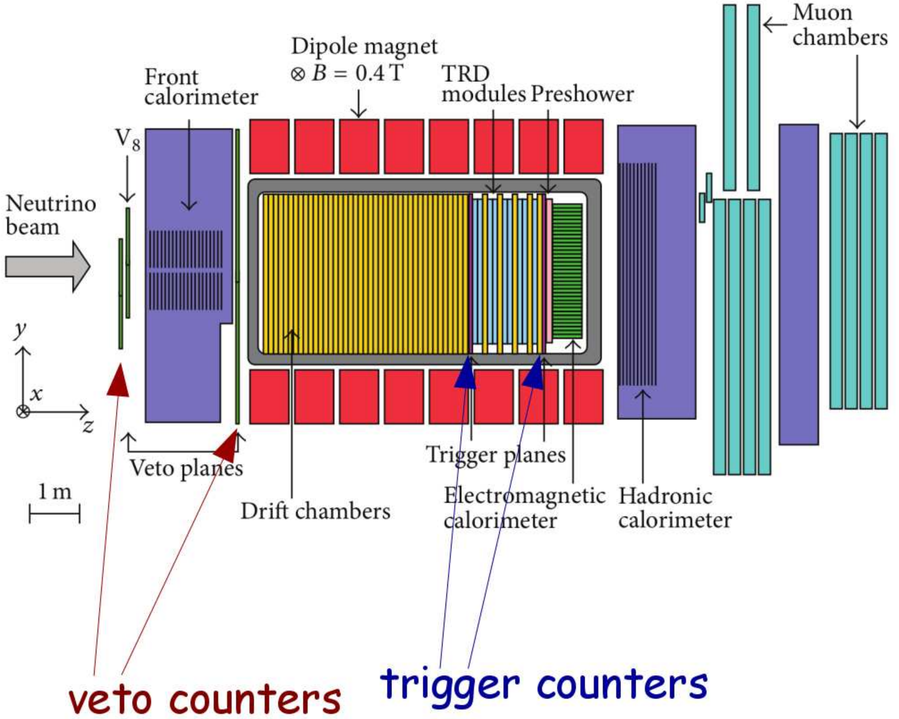
- e.g.: CERN LEP experiments
- complex detectors, moderate trigger rate, very little background
- little pileup, limited channel occupancy
- simpler, slow gas-based main trackers

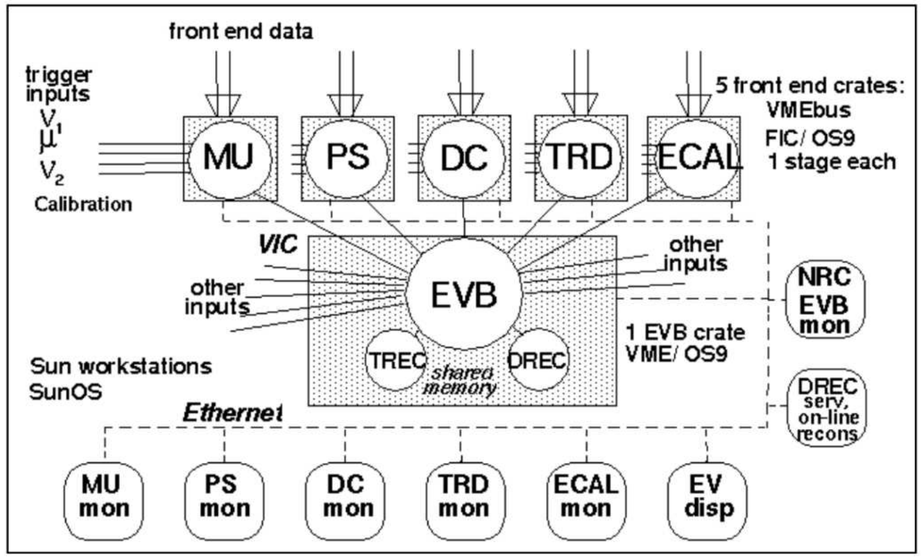


LEP
- 10^5 channels
- 22μs crossing rate
  - no event overlap
- single interaction

NOMAD (1995-1998)

- Search for $\nu_\mu \rightarrow \nu_\tau$ oscillations at the CERN WA neutrino facility (WANF)
- 2.4×2.4 m^2 fiducial (beam) area
- Two 4ms spills with 1.8×10^{13} P.o.T. each (ν spills)
- One (2s) slow-extraction spill (μ spill)
- 14.4s cycle duration





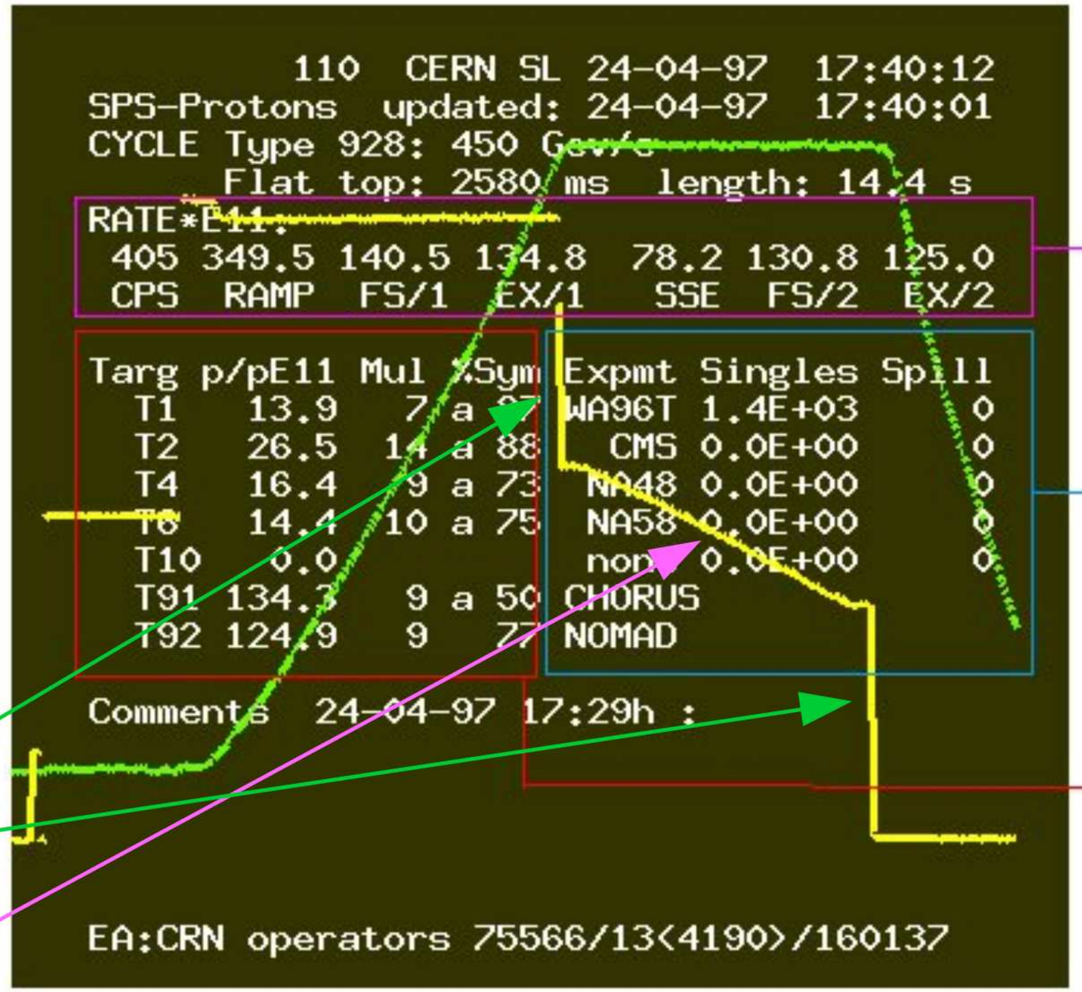
→ DAQ layout

WANF - SPS SuperCycle

14.4 s cycle length

2 × 4 ms neutrino spills (f/s extractions)

1 × 2 s muon spill (slow extraction)



Intensities in the SPS

Data from experiments

Steering on targets

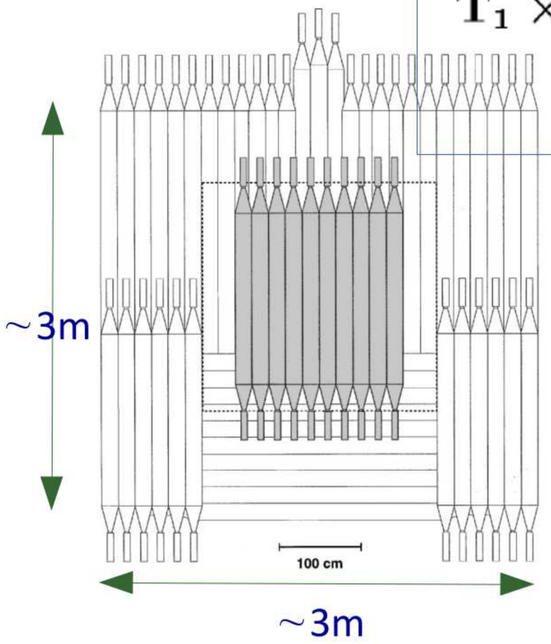
f/s extractions

slow extraction

Triggering once more ...

menu for NOMADs:

ν-spill triggers	μ-spill triggers
$\overline{\mathrm{V}} \times \mathrm{T}_1 \times \mathrm{T}_2$	$\mathrm{V} \times \mathrm{T}_1 \times \mathrm{T}_2$
$\overline{\mathrm{V}_8} \times \mathrm{FCAL}$	$\mathrm{V}_8 \times \mathrm{T}_2$
$\overline{\mathrm{V}_8} \times \mathrm{FCAL}' \times \mathrm{T}_1 \times \mathrm{T}_2$	$\mathrm{V}_8 \times \mathrm{T}_1$
$\overline{\mathrm{T}_1 \times \mathrm{T}_2} \times \mathrm{ECAL}, \overline{\mathrm{V}_8} \times \mathrm{ECAL}$	$\mathrm{V}_8 \times \mathrm{T}_1 \times \mathrm{T}_2 \times \mathrm{FCAL}'$
RANDOM	$\mathrm{V} \times \mathrm{T}_1 \times \mathrm{T}_2 \times \mathrm{ECAL}$

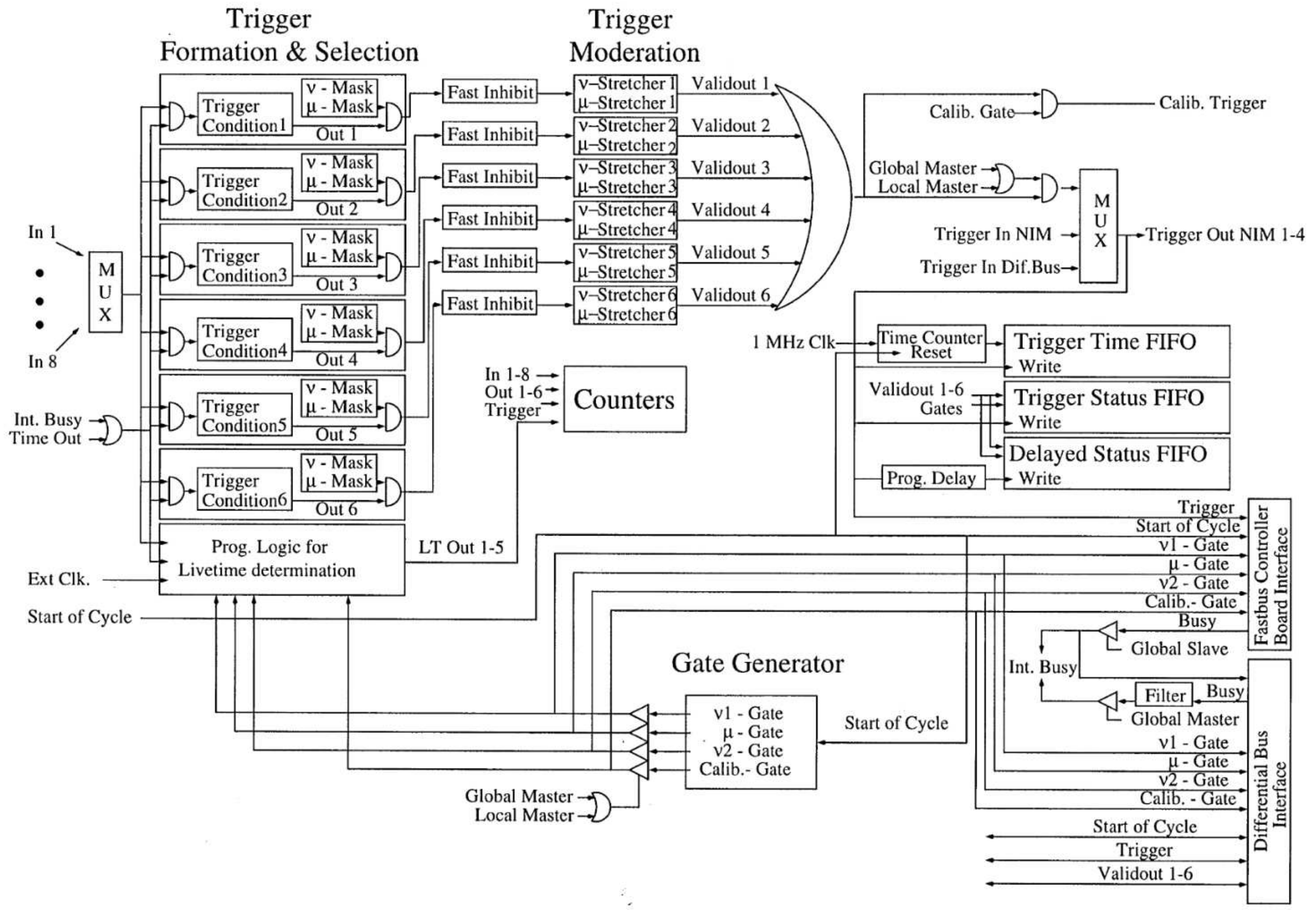


veto counters (central shaded area is V8)

Triggering → FPGA.s at work

MOdular TRIgger for NOmad (MOTRINO):

6 VME boards providing local and global trigger generation and propagation



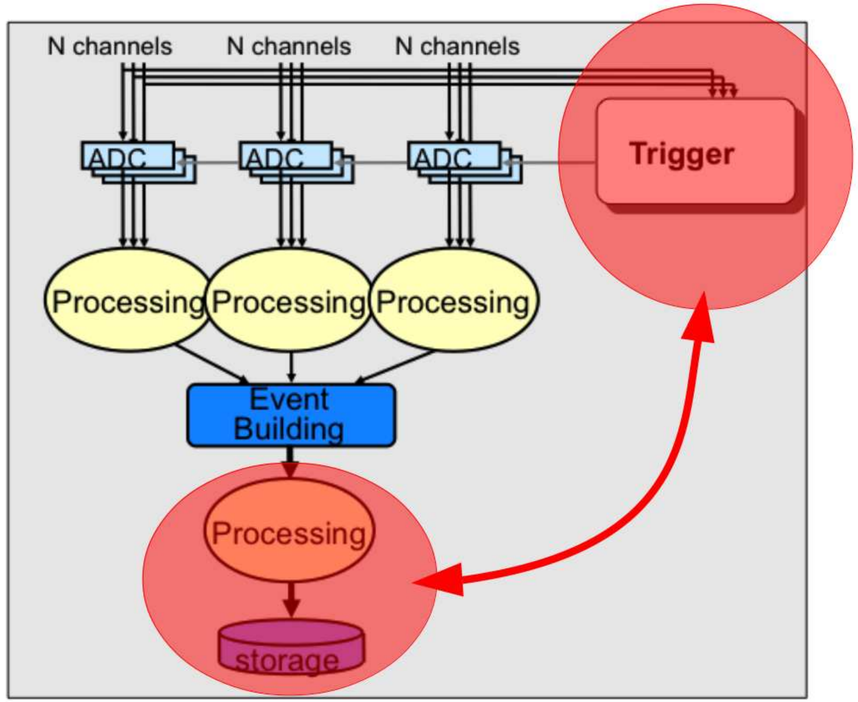
NOMAD DAQ

- ~30(?) (64 or 96 channel) Fastbus xDC [x=Q,P,T] boards
- Typically:
    - ~15 evts in each 4ms spill (neutrino triggers)
    - ~60 evts in each 2s spill (muon triggers)
    - 256-events in off-spill calibration cycles (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 done off-spill (i.e. off-beam)

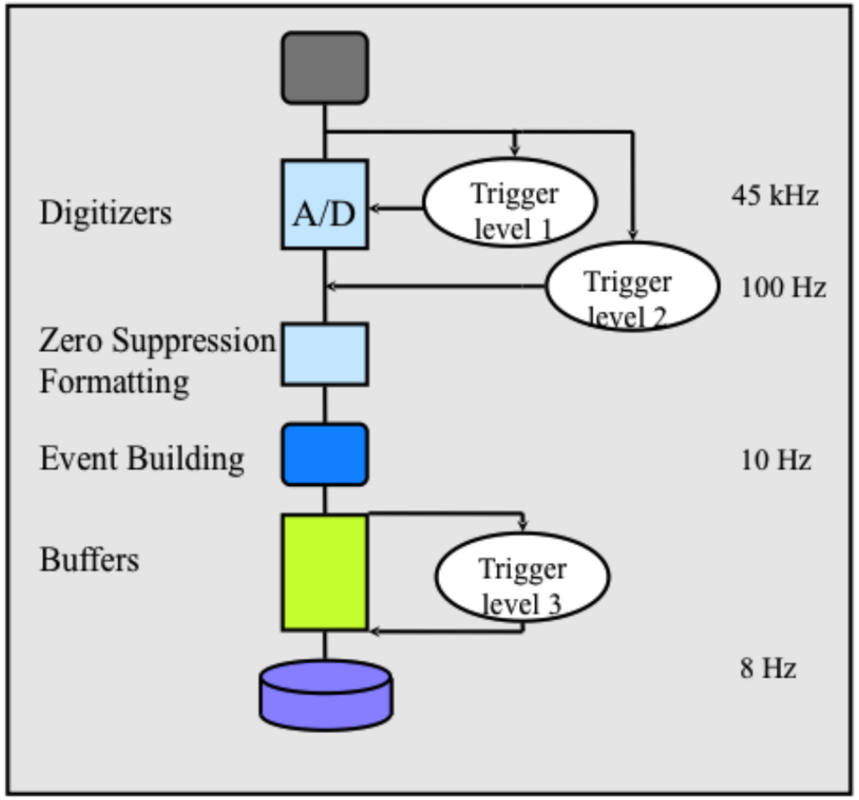
bottlenecks ?



- 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



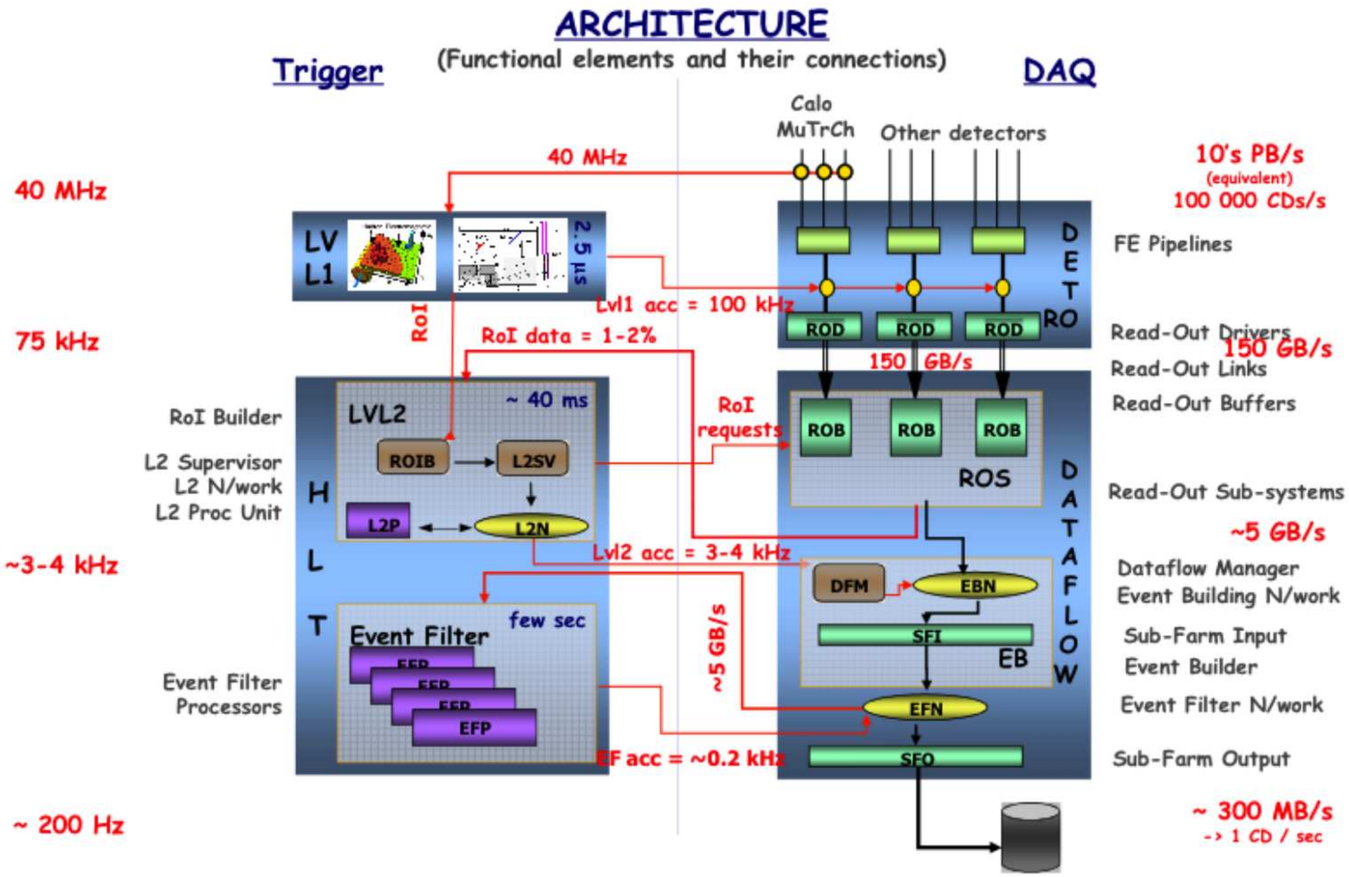
- more complex filters
- → slower
- → applied later in the chain

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!

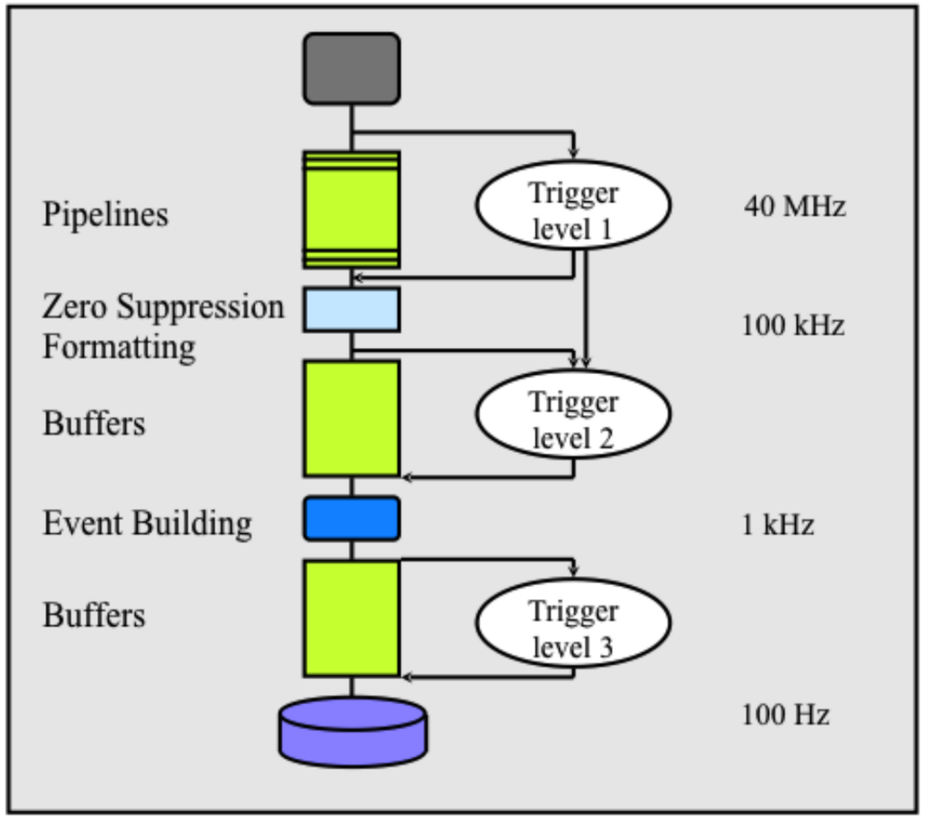
LHC
- 10^7 channels
- 25ns crossing rate
  - high event overlap
- 20 interactions
- L1 ~10^5 Hz
- L2 ~10^3 Hz
- L3 ~10^2 Hz
- 1MB/ev → 100MB/s



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

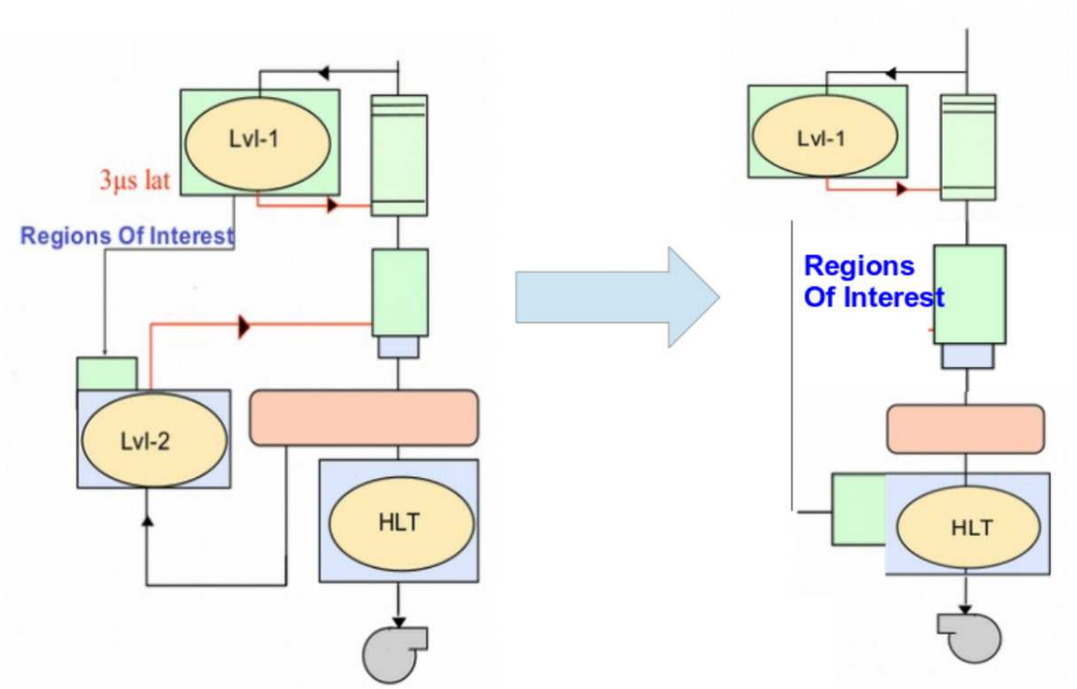
ATLAS Run-1 Architecture

- 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 the LHC experiments and upgrades" 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

Off-Topic: Event Selection Latency

- L1 : O(1 μs in real-time), let say = 2.5 μs
- L2 : O(10 ms) → let say = 40 ms
- L3(HLT) : O(s) → let say = 1 s

Q: do these 3 numbers mean the same thing ?

Latency and Real-Time

real time: system must respond within some fixed delay

→ *Latency = Max Latency*

→ over fluctuations bad, will create dead time

non-real-time: system responds as soon as it's available

→ *Latency = Mean Latency*

→ over fluctuations fine, shouldn't create dead time

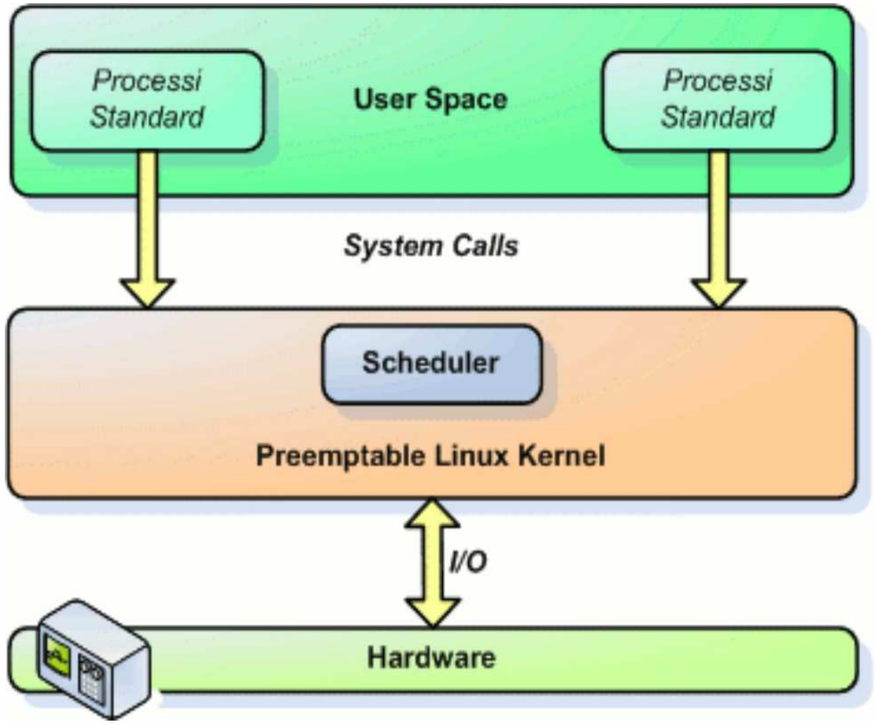
real time o.s. :

very stable time delay in responding to events

standard unix kernels are not real time:

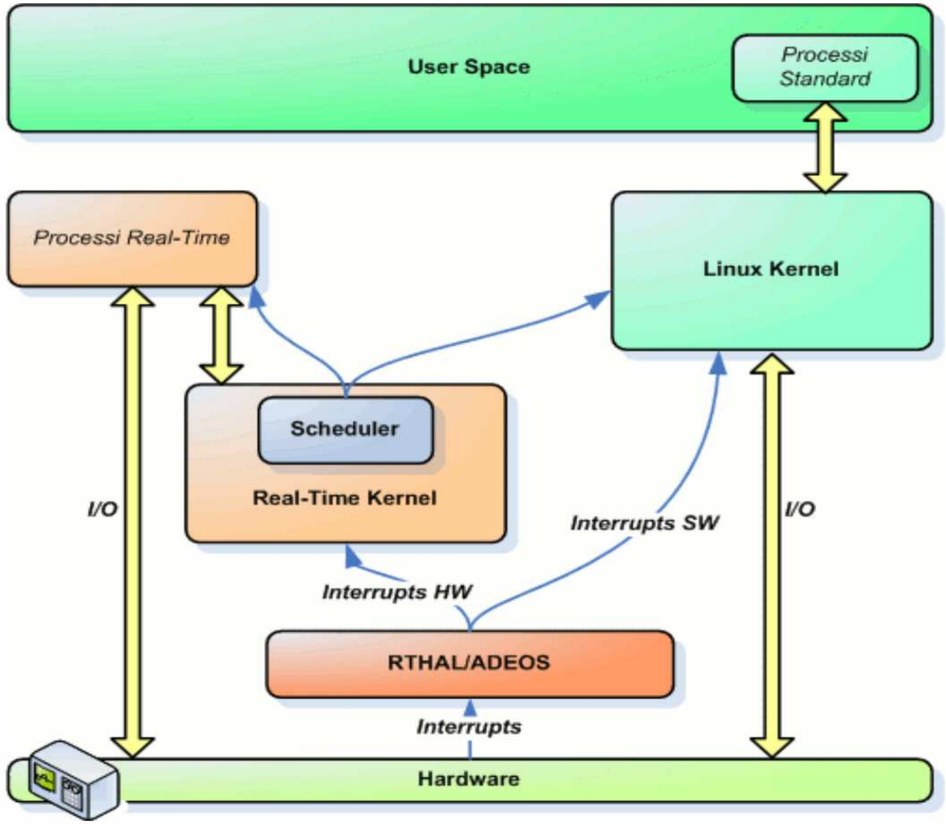
a system call can in principle take any time

Off-Topic: Real-Time Linux



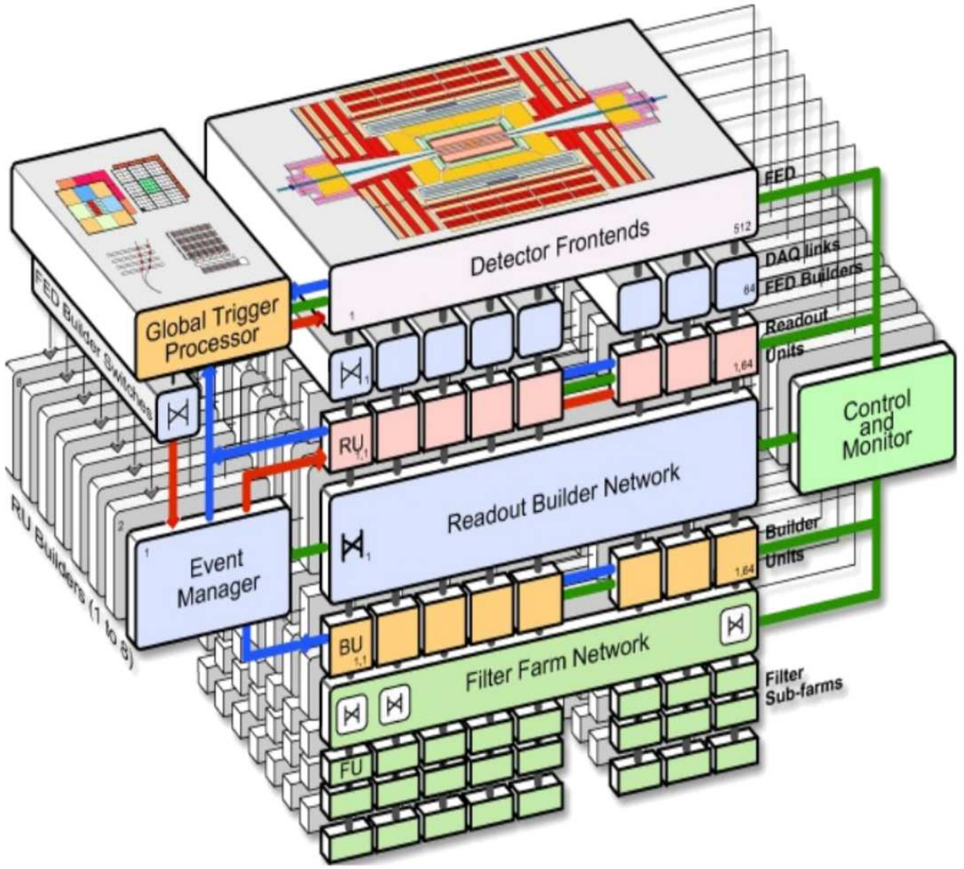
Low-latency patch (Ubuntu Studio):

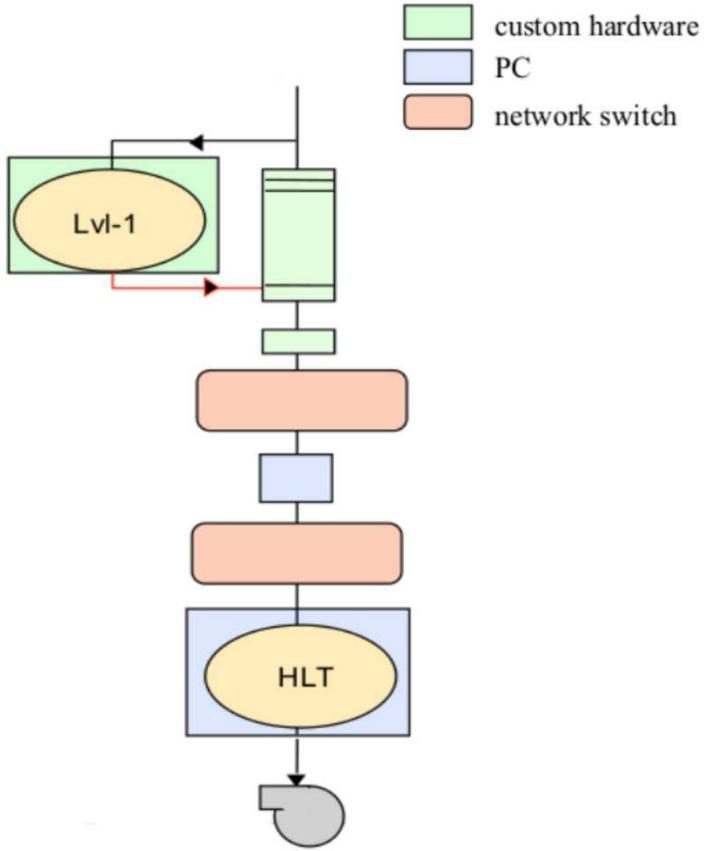
Interruptible linux kernel



RTAI: linux kernel runs as a higher priority application

CMS!



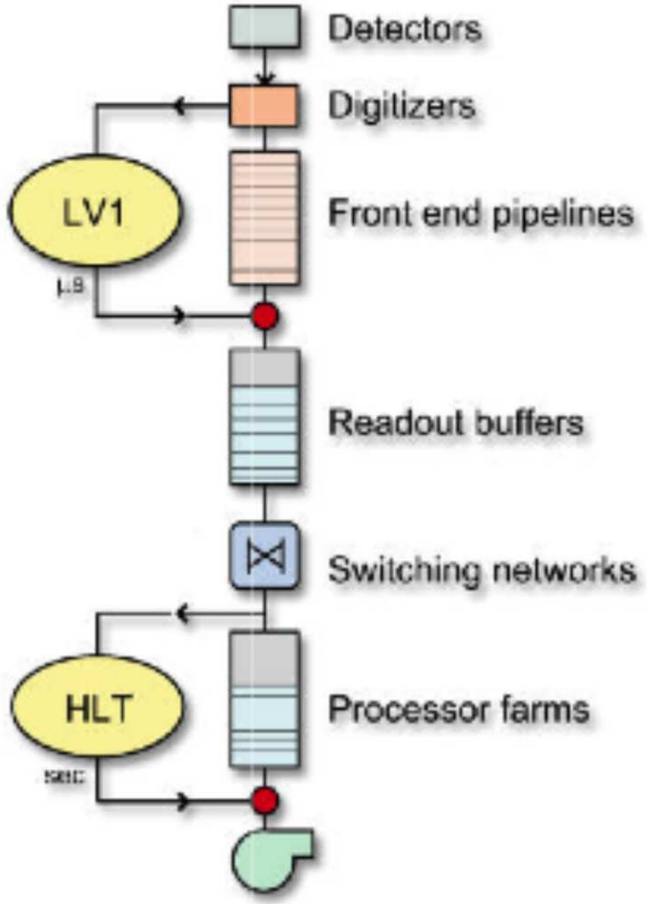


CMS TDAQ Design - S. Cittolin @ISOTDAQ 2010

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 Run 2

ATLAS:

more like CMS

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

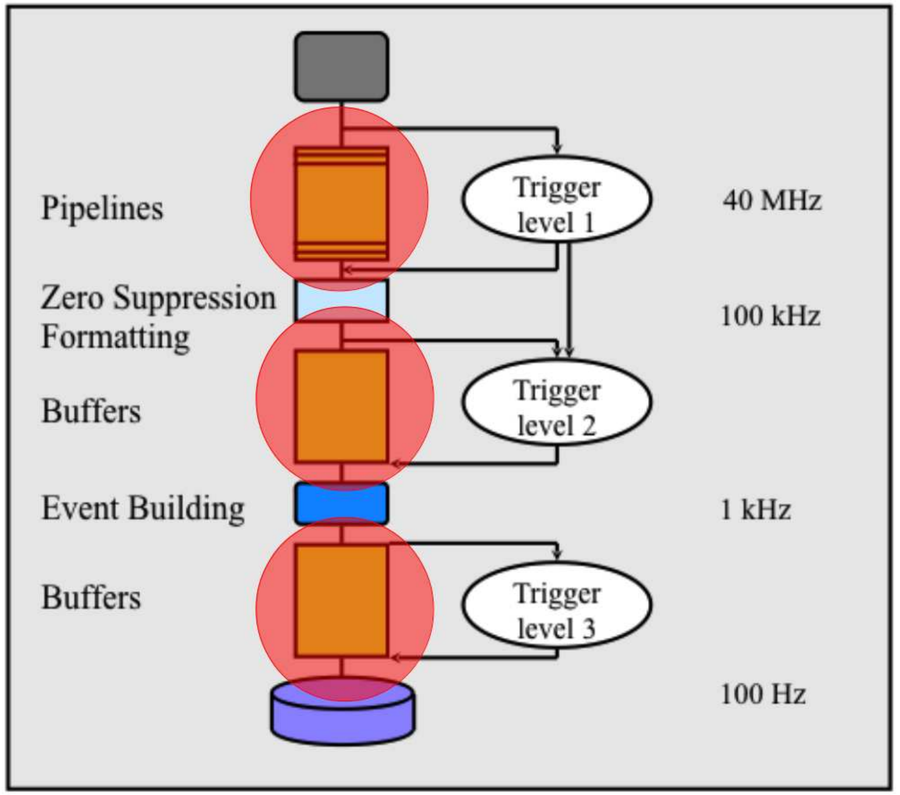
CMS:

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

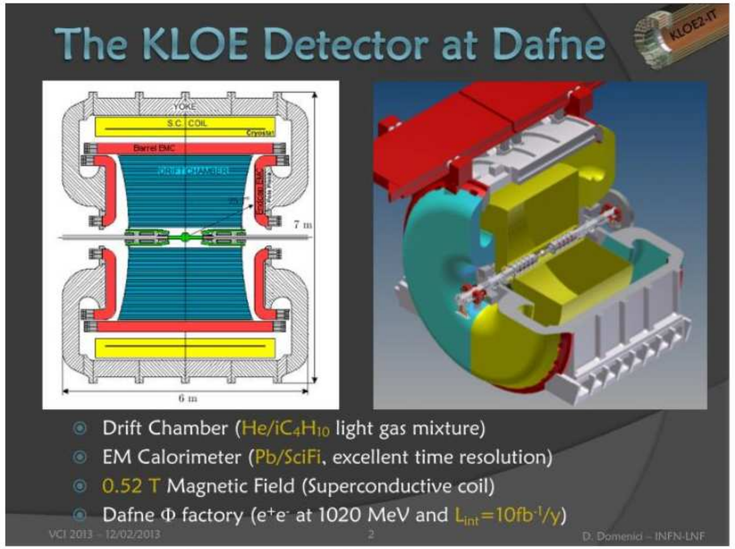


- Buffers are not the <final solution> they can overflow due to:
  - bursts
  - unusual event sizes
- Discard
  - local, or
  - "backpressure", tells lower levels to discard

Who controls the flow?
FE (*push*) or EB (*pull*)

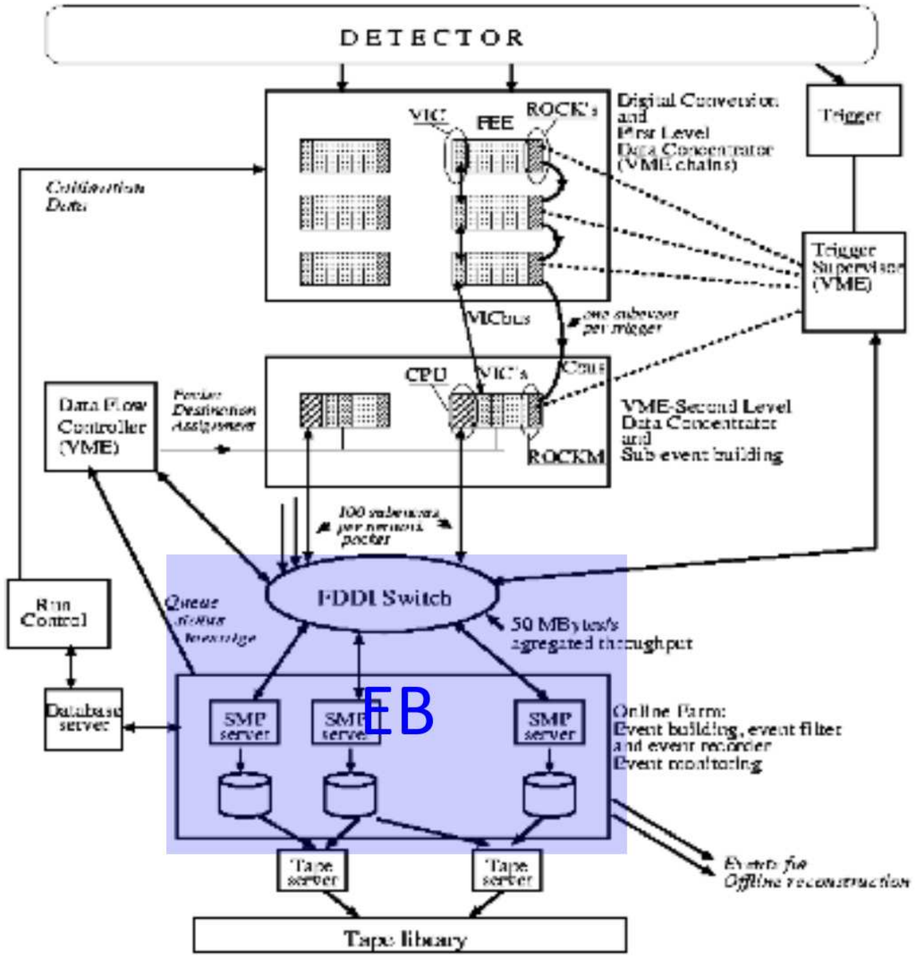
a *push* example: KLOE

- 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



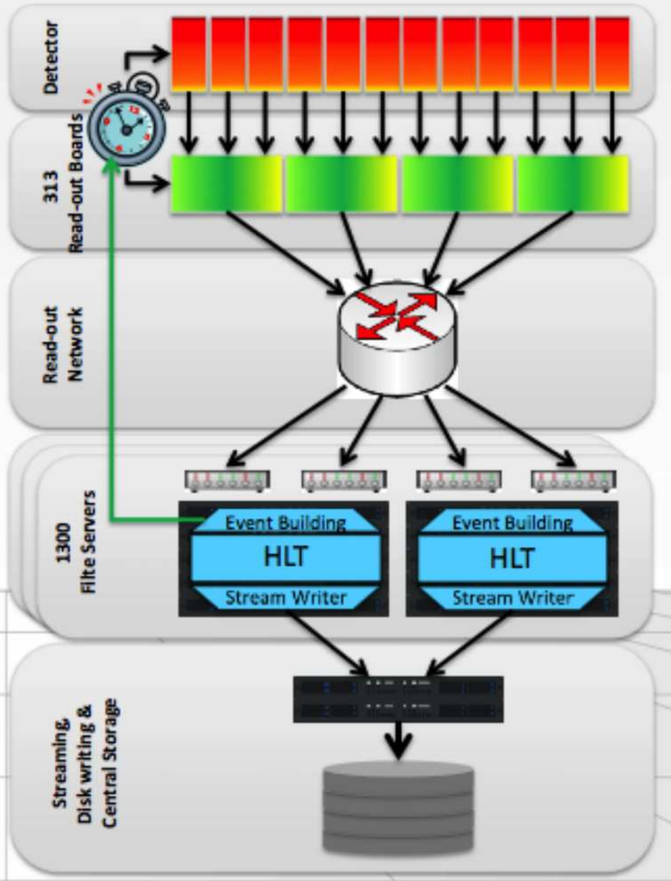
- deterministic FDDI network
- not real need for buffering at FE
- *push* architecture vs pull used in ATLAS
  see DAQ Software lecture
- try EB load redistribution before resorting to backpressure

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"

Looking forward to LS2 and beyond

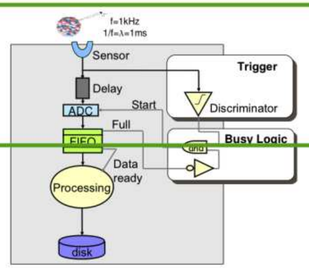
On some long term, all experiments looking forward to significant increase in L1 trigger rate and bandwidth. ALICE and LHCb will pioneer this path during LS2

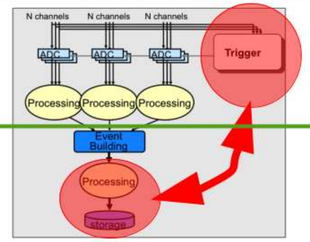
DAQ@LHC Workshop

- First level trigger for Pb-Pb interactions **500 Hz → 50 kHz**
- 22 MB/event
  - **1 TB/s readout → 500 PB/month**
- Data volume reduction
  - on-line full reconstruction
  - discard raw-data
- Combined DAQ/HLT/offline farm
  - COTS, FPGA and GPGPU

- **1 MHz → 40 MHz** readout and event building → trigger-less
  - trigger support for staged computing power deployment
- 100 kB/event
  - on-detector zero suppression → rad-hard FPGA
  - 4 TB/s event-building

Trends





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

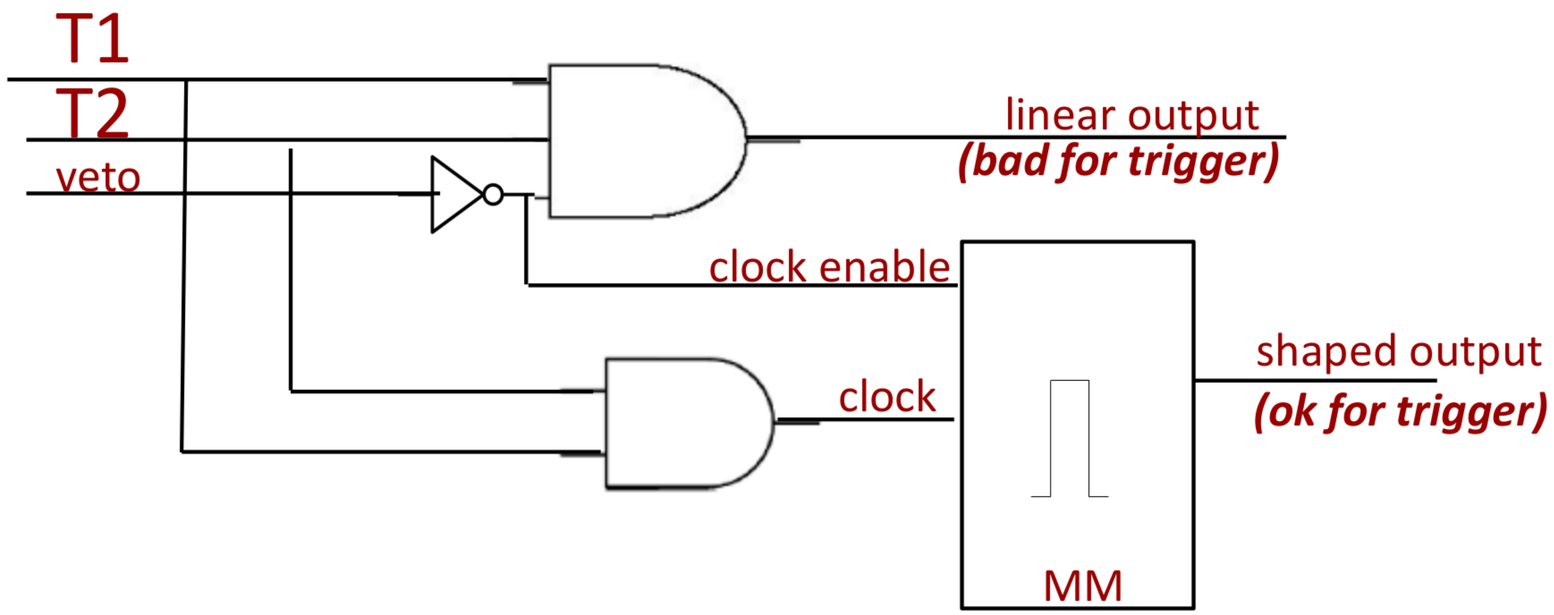
- Deal with dataflow instead of latency
- Use COTS network and processing
- Use “network” design already at small scale
  - easily get high performance with commercial components

take care, lot of issues not covered:

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

Lost & Found
(off-topics)

Appendix A: Trigger Coincidence



MM = Monostable Multivibrator = One-shot Pulse Generator

Exercise: Time Analysis

1) Linear Output = T1 and T2 and not Veto

2) Shaped Output = Fixed-duration Pulse at ((rising edge of (T1 and T2)) and not Veto)

1) Veto must completely overlap T1 and T2

Out-of-time veto may change the duration of (T1 and T2) coincidence

Early (out-of-time) veto may cause the (T1 and T2) coincidence rising edge to jitter

2) Veto must only overlap the rising edge of (T1 and T2) → very short in-time veto is required

Out-of-time veto may only veto the pulse generation but can't never produce shorter or jittering pulses

Appendix B: Profiling

Take care: optimize your code – first of all - where it really needs. To get it, you may use of profiling.

for C/C++ code, look (for example) at this gprof tutorial:

http://www.thegeekstuff.com/2012/08/gprof-tutorial/

Very simple, at least for standalone code ...

Appendix C: backtrace

Segfaulting ? Have a look at backtrace:

https://www.gnu.org/software/libc/manual/html_node/Backtraces.html

BACKTRACE(3) Linux Programmer's Manual BACKTRACE(3)

NAME

backtrace, backtrace_symbols, backtrace_symbols_fd - support for application self-debugging

SYNOPSIS

```
#include <execinfo.h>

int backtrace(void **buffer, int size);

char **backtrace_symbols(void *const *buffer, int size);

void backtrace_symbols_fd(void *const *buffer, int size, int fd);
```

HowTo

1) file “my_segf.cxx” : install a signal handler to print the backtrace

```
#include <stdio.h>
#include <execinfo.h>
#include <signal.h>
#include <stdlib.h>
#include <unistd.h>

void handler(int sig) {
  void *array[10];
  size_t size;

  // get void*'s for all entries on the stack
  size = backtrace(array, 10);

  // print out all the frames to stderr
  fprintf(stderr, "Error: signal %d:\n", sig);
  backtrace_symbols_fd(array, size, STDERR_FILENO);
  exit(1);
}

void baz() {
 int *foo = (int*)-1; // make a bad pointer
 printf("%d\n", *foo);       // causes segfault
}

void bar() { baz(); }
void foo() { bar(); }

int main(int argc, char **argv) {
  signal(SIGSEGV, handler);   // install our handler
  foo(); // this will call foo, bar, and baz.  Baz segfaults.
}
```

2) compile with -g debug flag on:

```
g++ -g -rdynamic my_segf.cxx -o my_segf
```

3) get the crash:

```
[roberto@bob-laptop ~]$ ./my_segf
Error: signal 11:
./my_segf(_Z7handleri+0x1c)[0x400a52]
/lib64/libc.so.6(+0x347e0)[0x7fa55f1c07e0]
./my_segf(_Z3bazv+0x14)[0x400aab]
./my_segf(_Z3barv+0x9)[0x400aca]
./my_segf(_Z3foov+0x9)[0x400ad6]
./my_segf(main+0x23)[0x400afc]
/lib64/libc.so.6(__libc_start_main+0xf1)[0x7fa55f1ac731]
./my_segf(_start+0x29)[0x400969]
```

4) crash is at (_Z3bazv+0x14) ... the function name is "_Z3bazv" (c++ function name mangling). How to get it ?

5) Demangle it thanks to: http://demangler.com/

6) Take the Answer: baz() → crash is at (baz+0x14)

7) crash is at (baz+0x14) ... open the debugger: gdb my_segf

```
(gdb) info address baz
Symbol "baz()" is a function at address 0x400a55.
```

8) so crash is at address (0x499a55+0x14) … then:

```
(gdb) info line *(0x400a55+0x14)
Line 24 of "my_segf.cxx" starts at address 0x400a65 <baz()+16>
  and ends at 0x400a7c <baz()+39>.
```

9) got it ! That’s not yet the reason but ...

Appendix D: Cables and Transmission Lines

Spoken about signals, amp.s, digitisers, … but ...

… almost nothing about how signals are transmitted over long distances. ***Is there any issue ?***

Q(1): what is a cable (for a single signal) ?

a couple of ideal conductors (R=C=L=0) ?

Q(2): which speed can it reach ?

Q(3): what's its impedance ?

Q(4): what does it to your signal ?

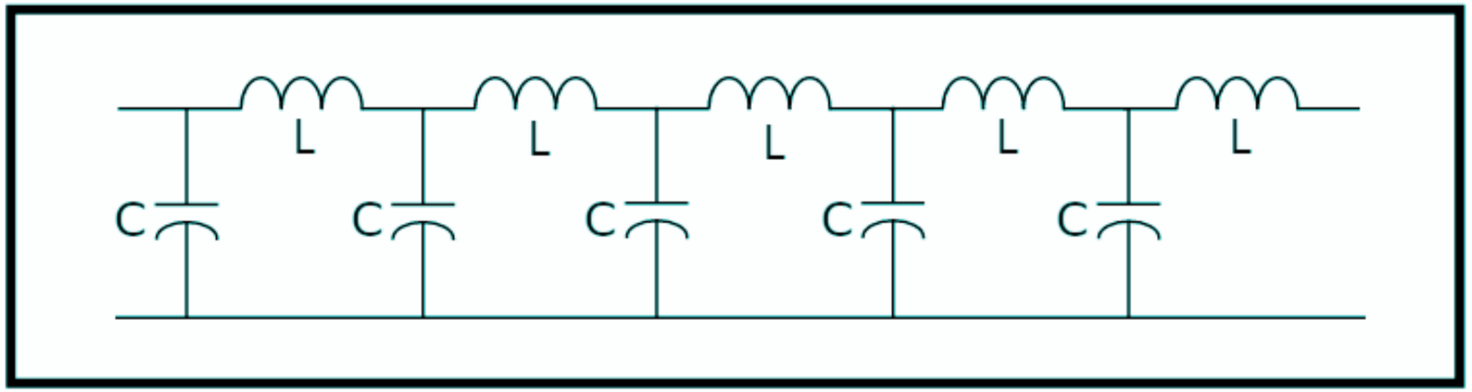
Ok the full line must be properly matched:

Z(out) = Z(cable) = Z(in)

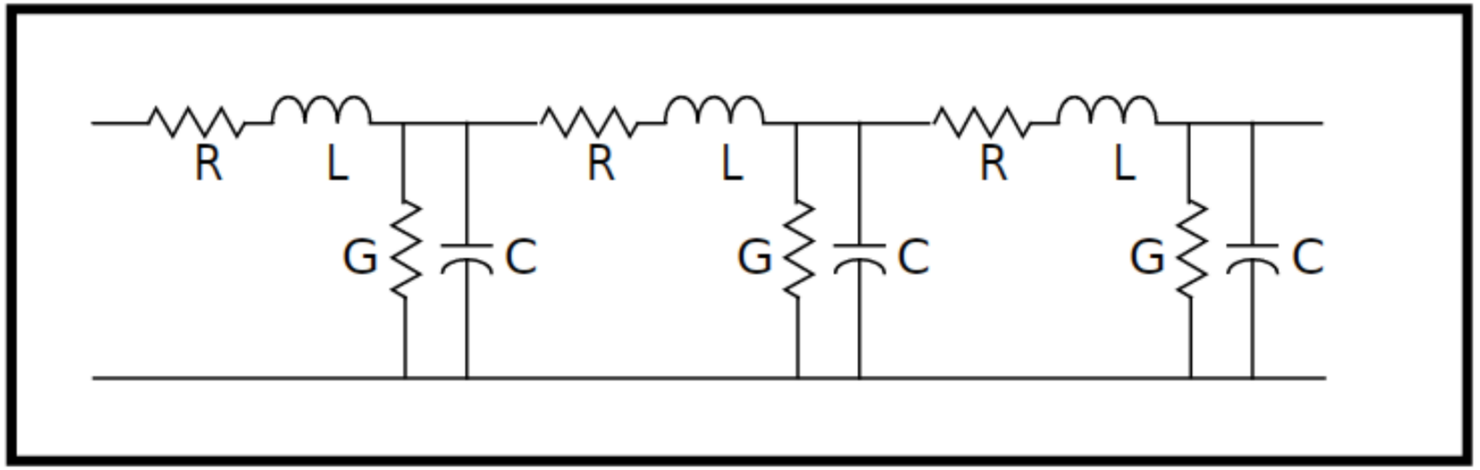
That's all ?

Cables and Transmission Lines

Lossless transmission line:



Lossy transmission line:

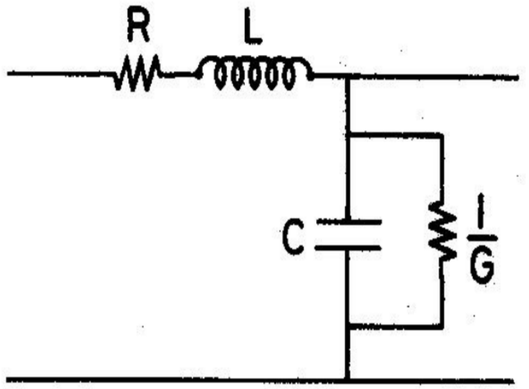


Cables

Cable element (dz):

$$L \approx \frac{\mu}{2\pi} \ln\left(\frac{b}{a}\right) \quad [\mathrm{H/m}]$$

$$C \approx \frac{2\pi\varepsilon}{\ln(b/a)} \quad [\mathrm{F/m}]$$



R depends on the frequency (skin effect)

G should be negligible

$Z = (L/C)^{1/2}$

$v_p = (LC)^{-1/2} = (\mu\varepsilon)^{-1/2}$

Cables

Equation for standing waves:

$$\frac{\partial^2 V}{\partial z^2} = LC\frac{\partial^2 V}{\partial t^2} + (LG + RC)\frac{\partial V}{\partial t} + RGV$$

solution:

$$\frac{d^2 V}{dz^2} = (R + i\omega L)(G + i\omega C)V = \gamma^2 V$$

$$\gamma = \alpha + ik = \sqrt{(R + i\omega L)(G + i\omega C)}$$

R usually dominated by the skin effect:

$$R(\omega) = r*D/(4*\delta)$$

r = resistance per unit length
D = diameter internal conductor
δ = skin depth ~ $1/\sqrt{\omega}$

Cable Losses

Neglecting the transconductance G:

$$\alpha = R(\omega)/(2Z_0) \sim c\sqrt{\omega}$$
$$k = \omega\sqrt{RC} = \omega/(\beta c)$$
$$V(z,t) = V_1 \exp(-\alpha z)\exp[i(\omega t - kz)]$$

50-Ohm fast (v = 4 ns/m) CERN-store cables:

04.61.11.F - COAXIAL CABLE 50 OHM - TYPE C-50-6-1
04.61.11.H - COAXIAL CABLE 50 OHM - LOW LOSS - TYPE C-50-11-1

f(-3db, 40 m, cable C-50-6-1) ~ 120 MHz
f(-3dB, 40 m, low loss cable) ~ 640 MHz

Signal Distortions

Time parameter:

$$\alpha \sim \mu \sqrt{f}$$

$$\tau_0 = (\mu z)^2 / \pi$$

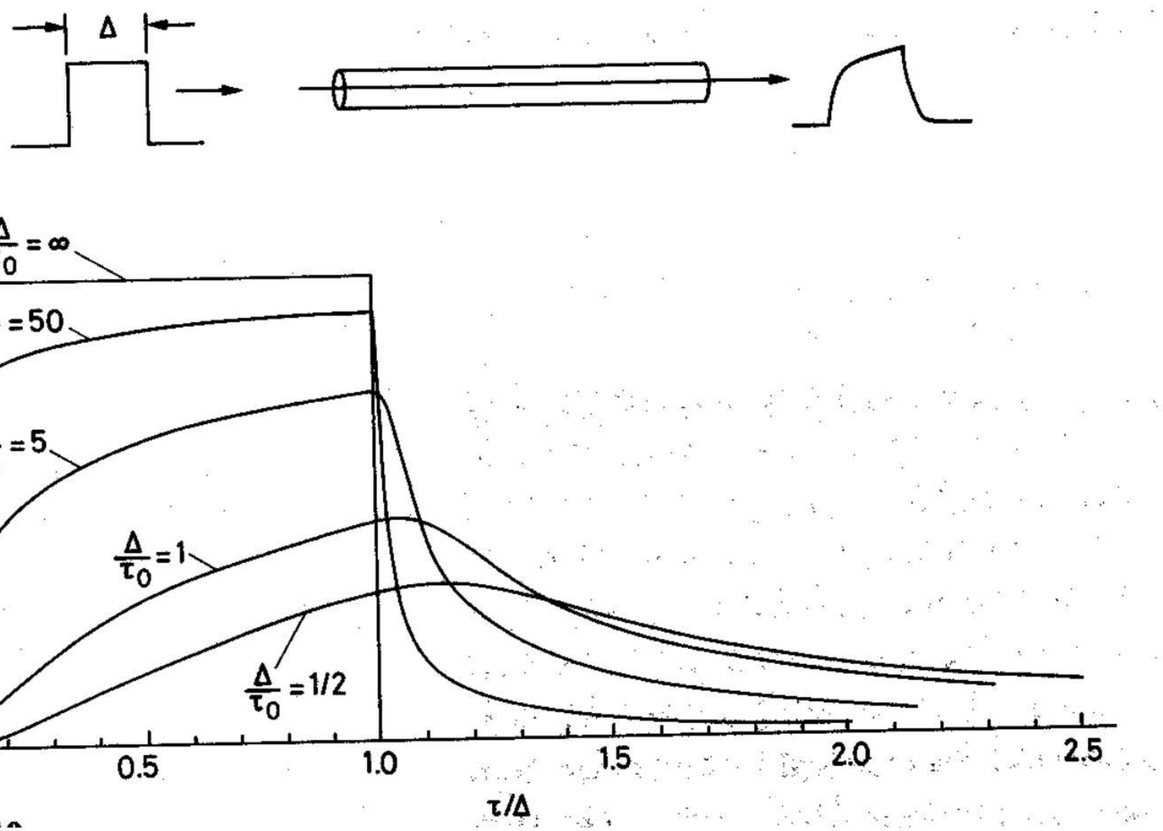
μz ~ 32*E-6 (C-50-6-1), 14E-6 (low loss cables)

$$\tau_0 \sim 320\, ns\, (C - 50 - 6 - 1)$$

$$\tau_0 \sim 60\, ns\, (low\ loss\ cables)$$

*** Take care: would like $\tau_0 \ll \tau$(signal)

Digital Pulse Distortions



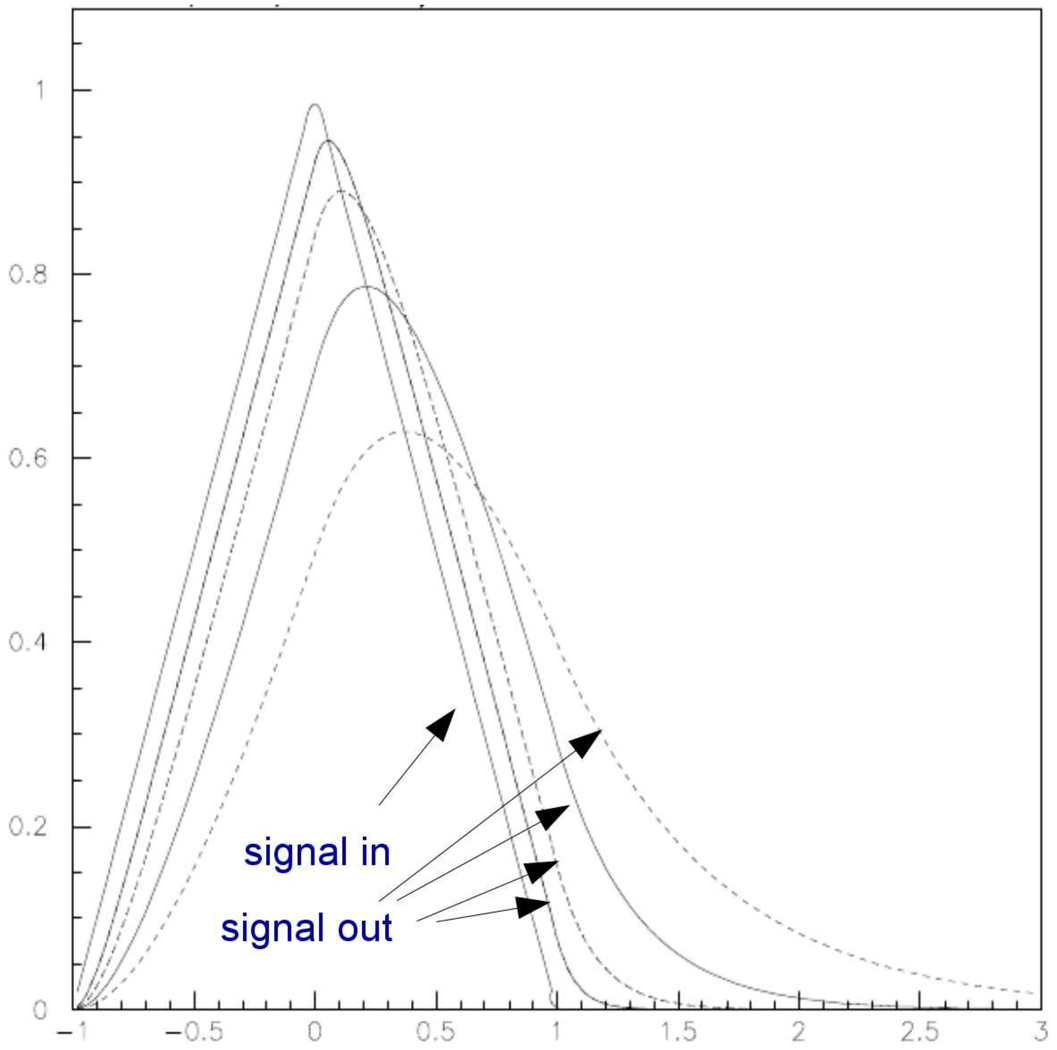
$$\text{for } \alpha \sim a\sqrt{f}$$

$$\tau_0 = (a\,z)^2/\pi$$

Bandwidth Effects – Analog Signals

~1ns analog-signal response for

BW ~ 300, 150, 75, ... MHz



Thank you for your patience …