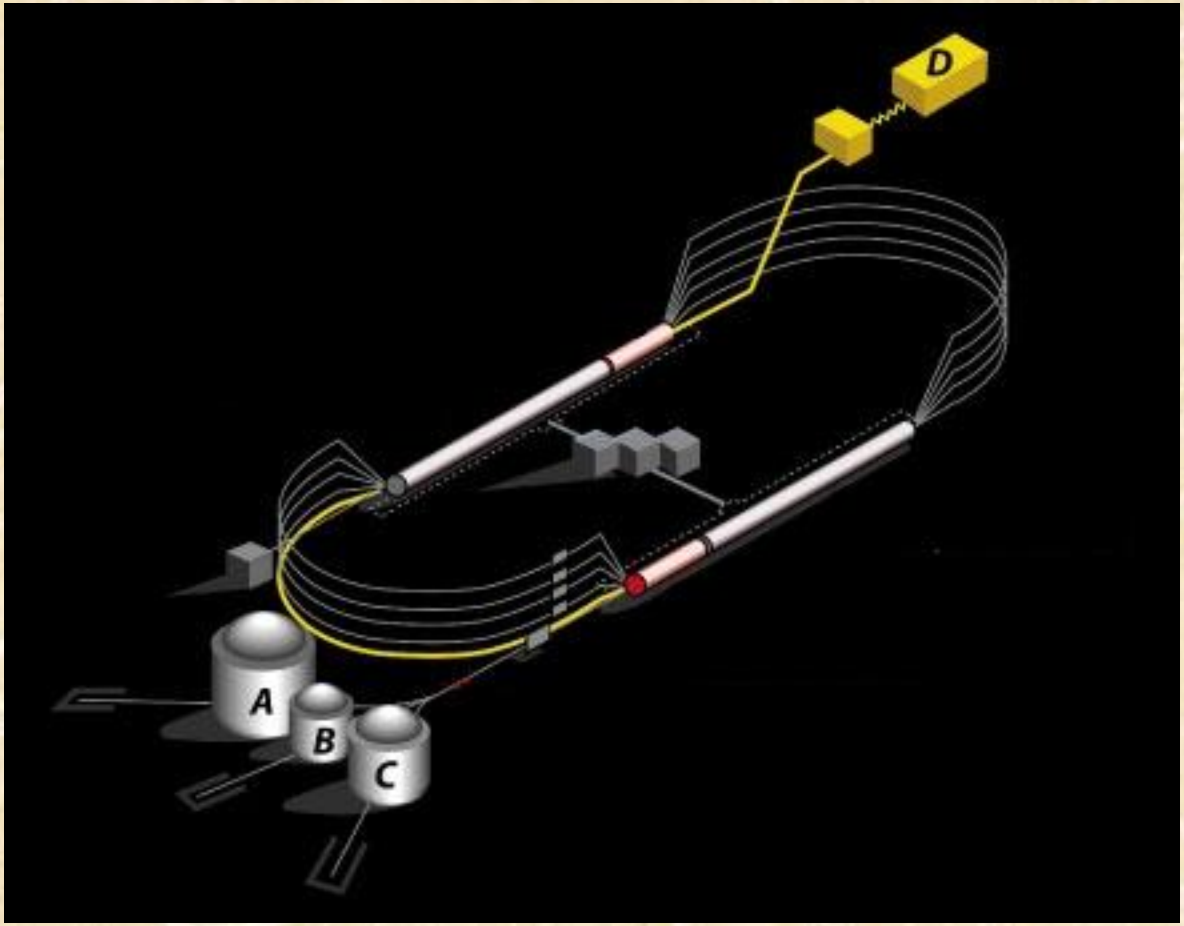
Data Acquisition Systems for Experimental Nuclear Physics



by David Abbott

Introduction

About me:

- Ph.D UNC Chapel Hill (1990)
- JLAB 1991-now
- With the Data Acquisition Group since 1994

About us:

- 6 member Physics support group
- Cebaf Center: F-Wing

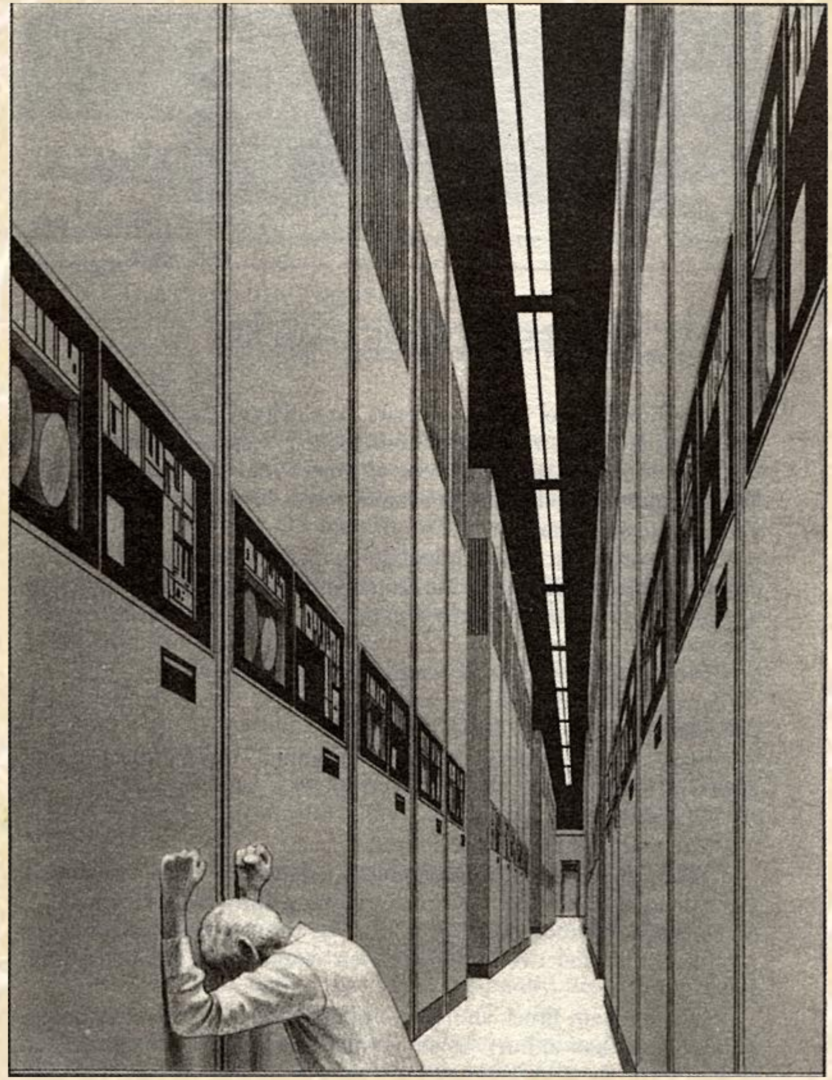
We’re in Here

Lab on the 1st Floor

What is this talk about...

- What is Data Acquisition?
- The anatomy of a DAQ system.
  - DAQ architectures.
  - What do all the bits and pieces do?
  - What does it all look like?
- What are we doing here at JLAB?
- Show and Tell along the way…

What is Data Acquisition?



What is the goal?

- The aim of a nuclear physics experiment is to gather data about nuclear interactions.
- Nuclear particles pass through detectors which generate electrical *signals*.
- These signals contain information about the particles - *type, energy, trajectory*.
- The data acquisition system *digitizes*, *formats* and *stores* this information in a way which can be retrieved for later analysis.

What are the problems?

- The complete set of signals which describe a single nuclear interaction is called an *Event*.
- There can be thousands to millions of events occurring per second.
- Detectors are large and distributed - containing many thousands of individual *channels*.
- Events are different sizes.
- Events occur at random.
- Only a few events are interesting.

Data Acquisition Requirements

- Move the data: Detector --> Storage
- Configure and control experiments
- Monitor data flow
- Monitor detectors/hardware
- Inform operator of problems
- Experiments can run for days/weeks/months…

The Anatomy of a DAQ System.

- **Triggering** (choosing events we want)
- **Readout** (digitizing detector signals)
- **Event formatting** (standardize what we’re saving)
- **Event building** (putting fragments together)
- **Event transport** (make events available to all)
- **Event storage** (save data for analysis)
- **Run Control** (configure-start-stop experiments)
- **Monitoring** (tell me what’s going on)
- **Slow Controls** (What is the other hardware doing?)

A DAQ System example

- Digital Camera is a "simple" physics DAQ system.
- 3-6 million channels
- *Dead-time* is important!
  - How long before I can take another picture??
- DAQ requires a "Real-time" response.

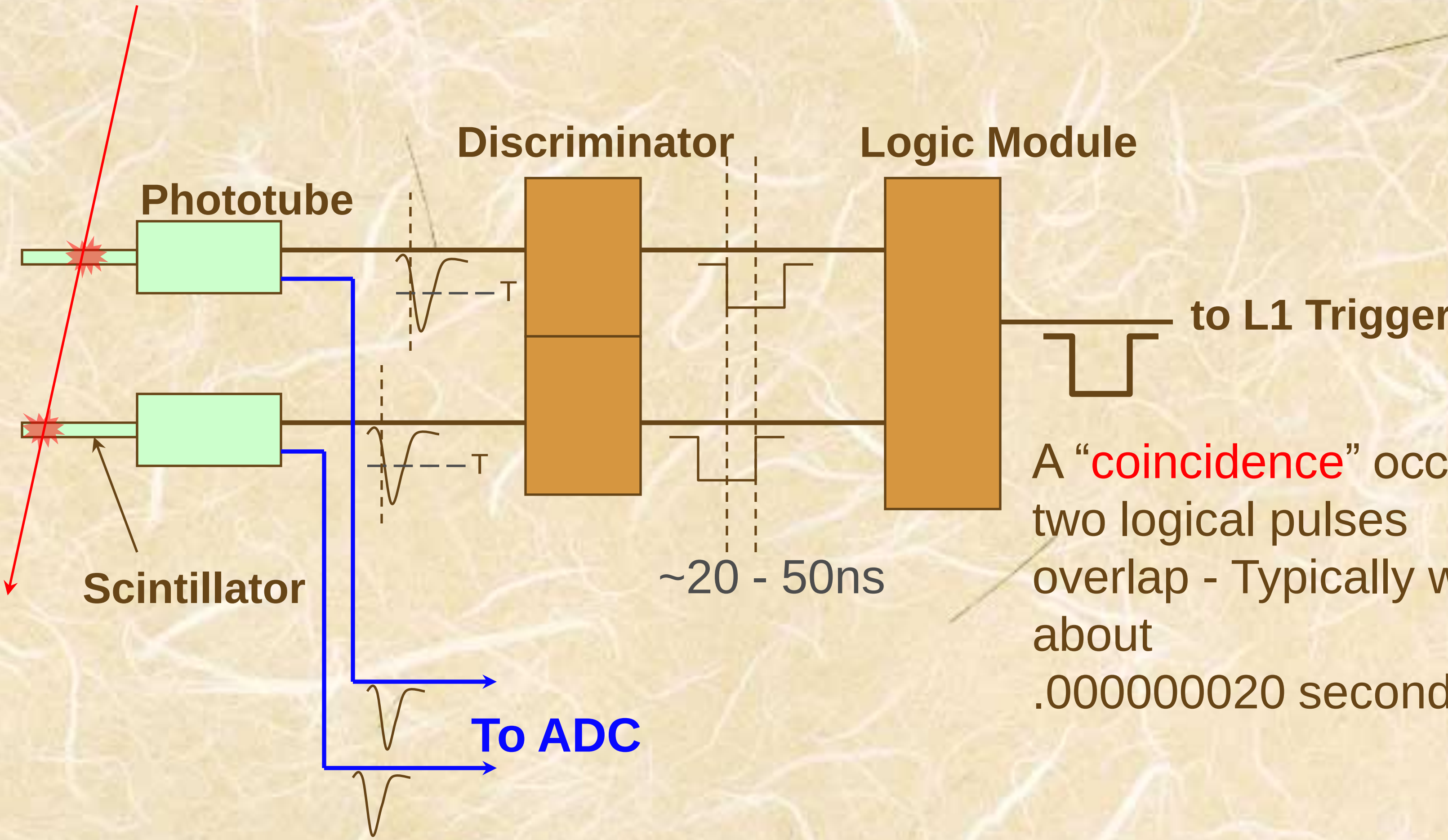
Triggering

- The data acquisition system needs to know when an interaction "*Event*" has occurred in the detector.
- Some detectors are faster than others.
- Signals from fast detectors are combined to make a decision on when an event has occurred. This is called a *trigger*.

Triggering cont…

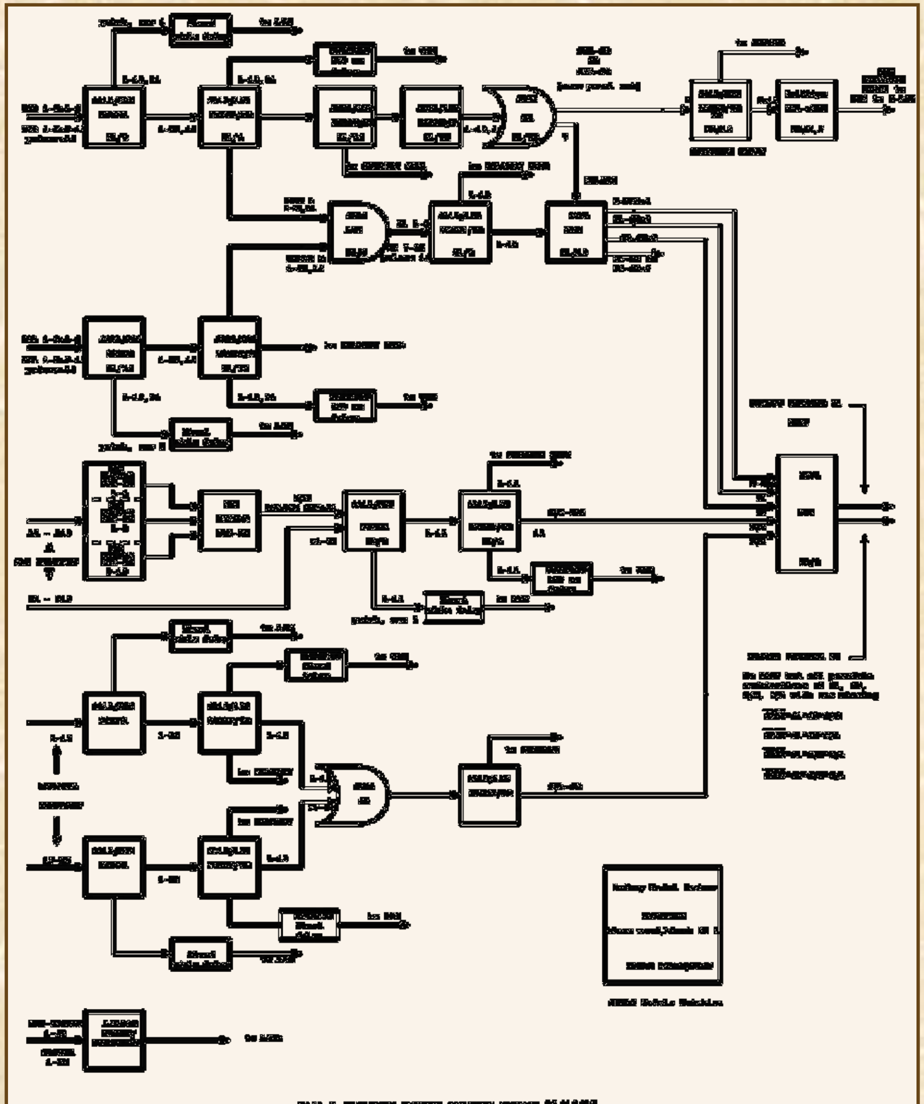
- Triggering serves two functions:
  - Tells the rest of the system when to read.
    - Trigger tells DAQ to read out data
    - DAQ tells trigger when it is busy
    - Busy time is called dead-time, and is minimized by a well designed DAQ architecture (see later).
  - Filters unwanted events.
    - Most triggers work in levels.
      - Level 1 is based on fast detectors like scintillators.
      - Level 2 is based on slower detectors (drift chambers).
      - Level 3 is usually a *software filter*.

A Simple Trigger



A “coincidence” occurs if two logical pulses overlap - Typically within about .000000020 seconds.

Hall A Trigger



What it really looks like.

Hall B CLAS Trigger

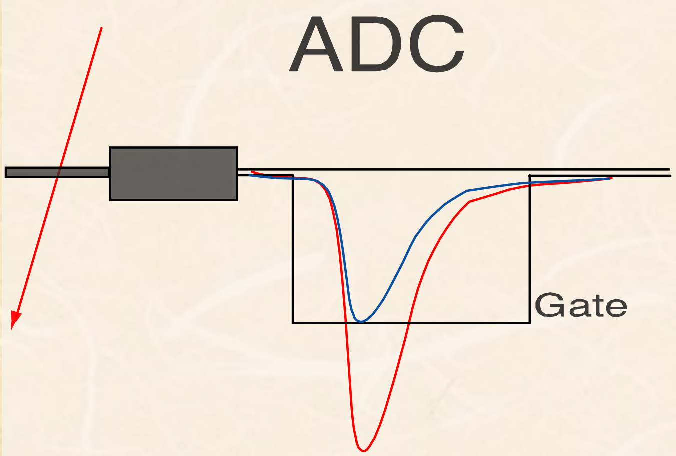
Readout

- Data takes the form of electrical signals.
  - Convert Analog --> Digital
  - Times - Time to Digital Converter, TDC
  - Voltages - Analog to Digital converter, ADC
  - Counts - scalars
- There are lots of signals spread over a large detector.
  - Modular readout duplicated many times
  - Plug-in modules require something to plug into so that they can all be accessed together --> ***Buses***

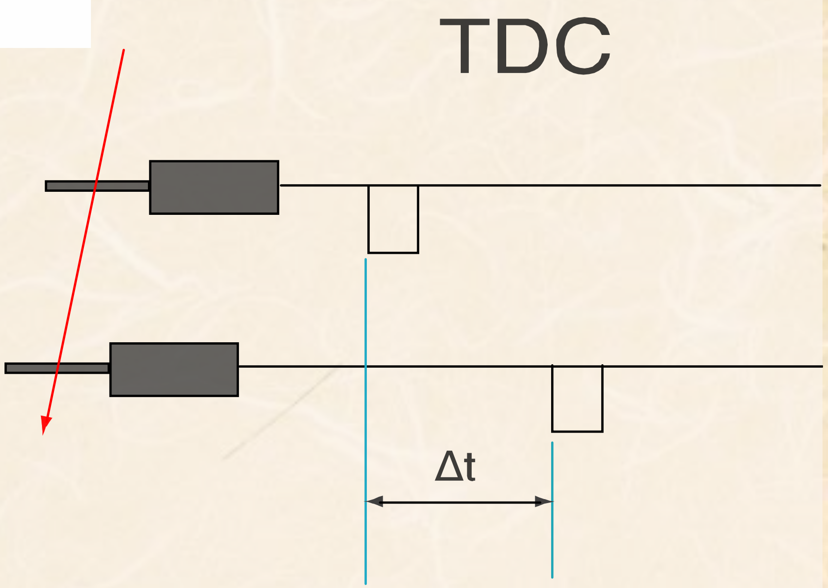
Digitizing Data

Detector data must be digitized to be stored on a computer. When a particle passes through a detector, we can measure the amount of energy (charge) it leaves (ADC), or we can measure the time it takes to travel between two different detectors (TDC).

The digitizer makes a 12bit number (0 - 4095) that is proprtional to the amount of charge (or time). The physicist must calibrate to digitizer so that he knows the energy (or seconds) each number represents.



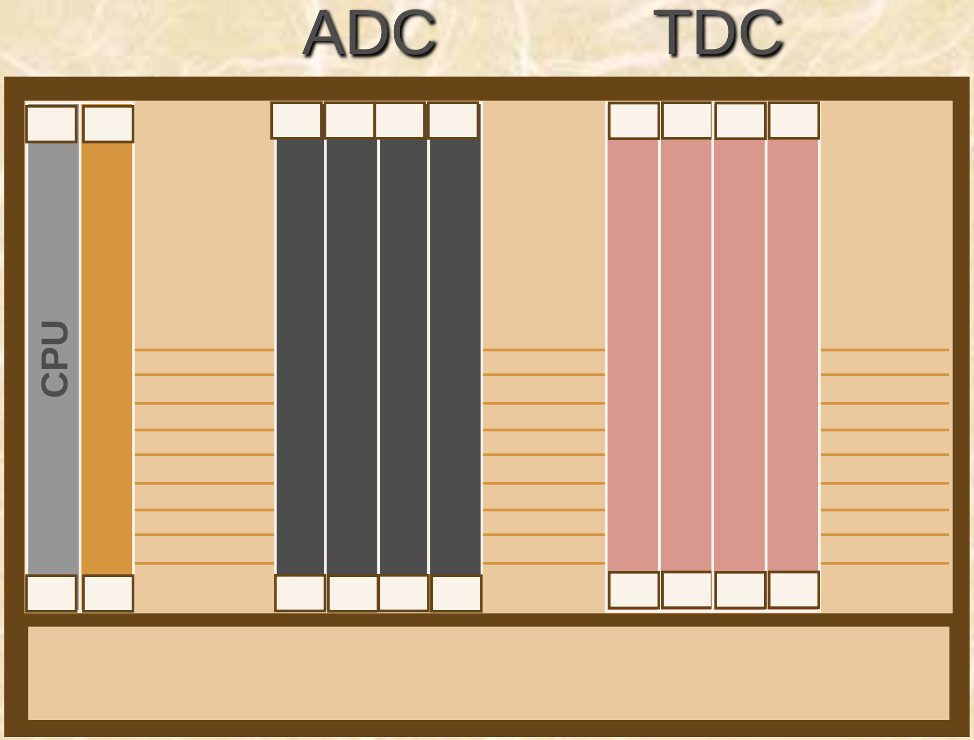
Digitized Value		Energy
1520	=>	50 MeV
3040	=>	100 MeV



Digitized Value		Time
220	=>	5 nanoseconds (.000000005 seconds)

Modules/Buses

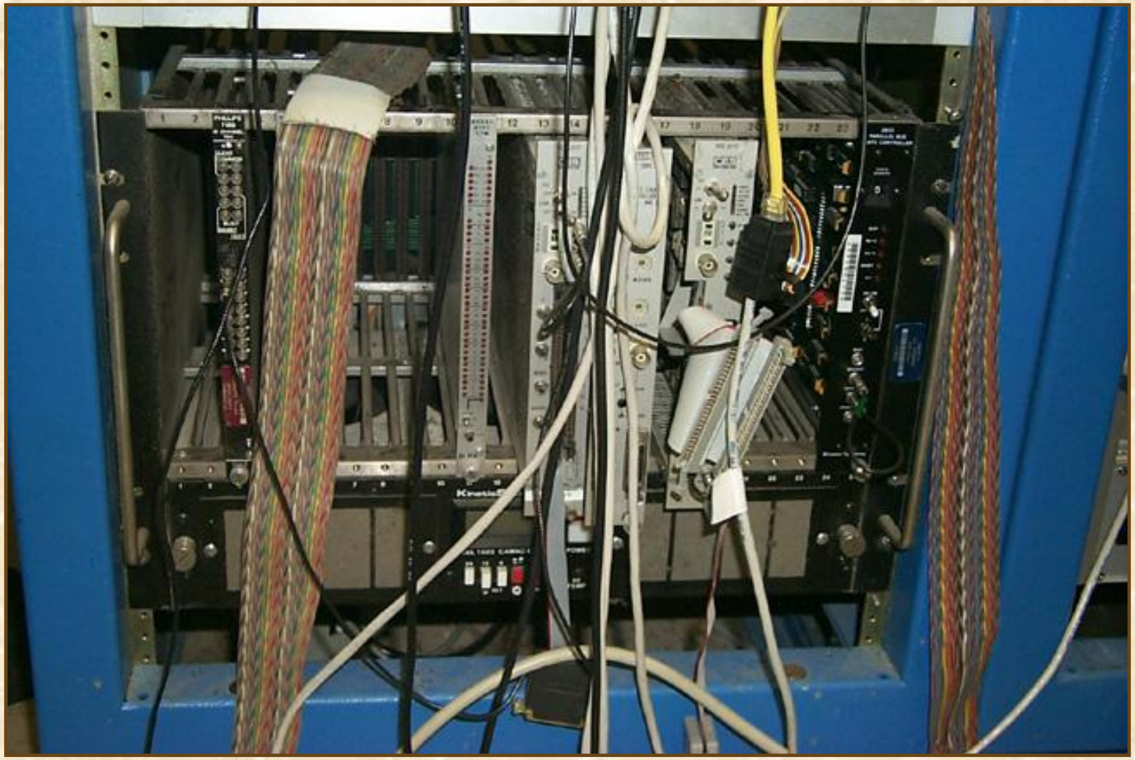
- Detectors in the Experimental Halls have many thousands of channels. Each Channel is read (digitized) by an ADC or TDC.
  - Pack many circuits onto one board.
  - Pack many boards into a box.
  - Pack the boxes into racks.
  - Use a standard bus to link everything.
- Standards: CAMAC, FASTBUS VME and PCI.



CAMAC

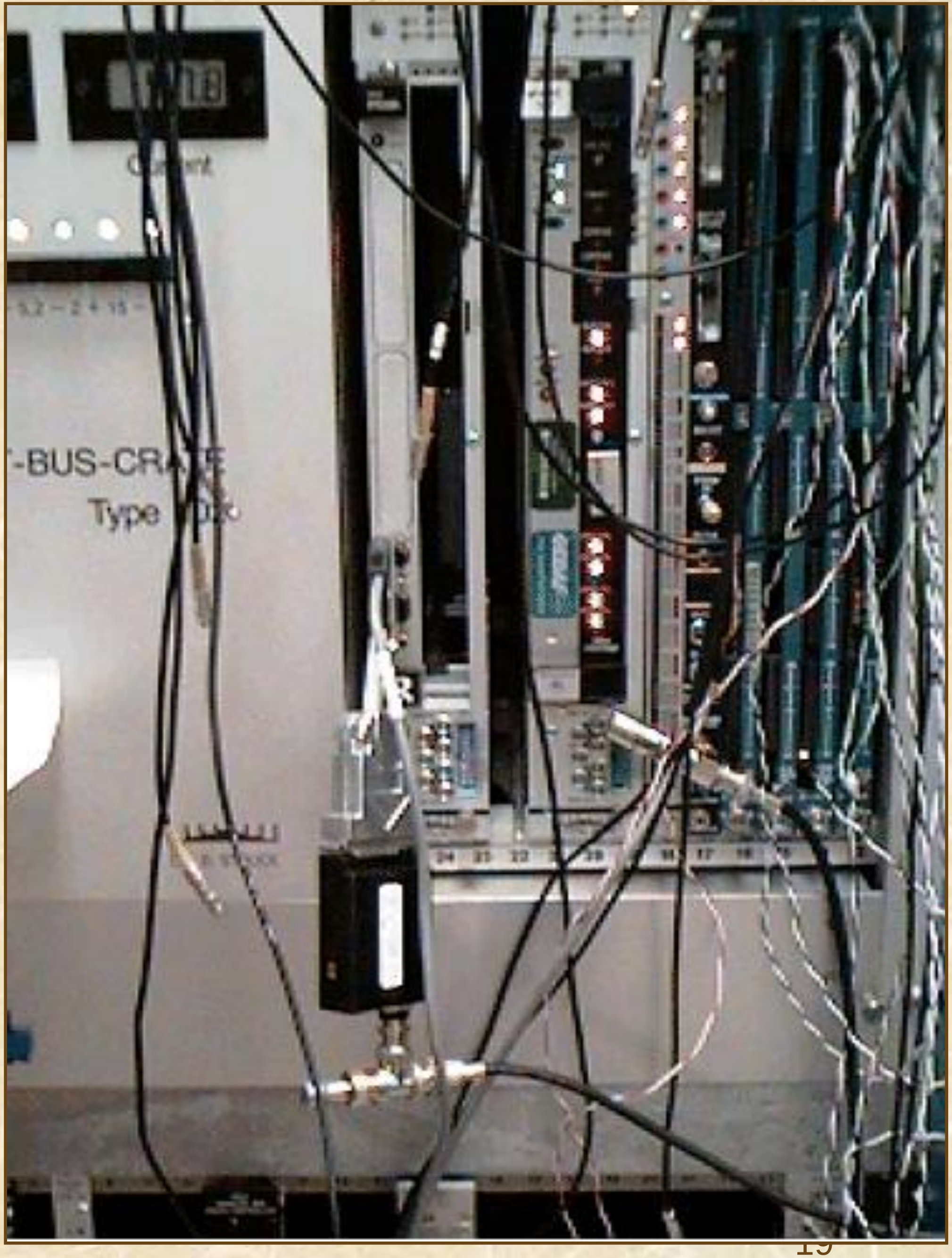
(Computer Automated Monitoring And Control)

- Old IEEE Standard
- 24 bit bus.
- Relatively slow
  (3 MB/sec).
- Small boards.
- A lot still around.
- Slow controls

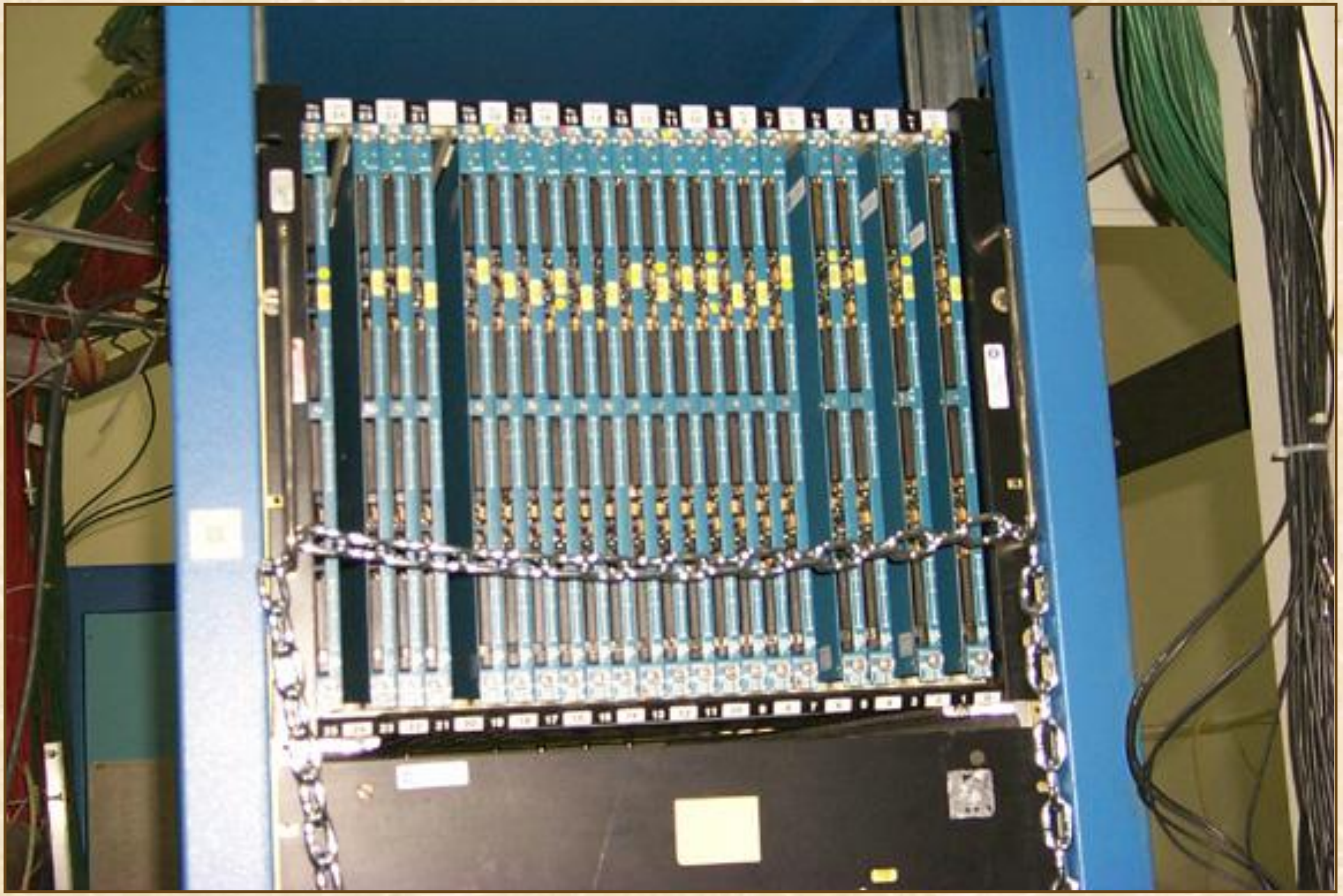


FASTBUS

- Designed by physicists for physicists.
- Large form-factor, high channel densities.
- 32 bit bus (40 MB/sec)
- Majority of JLAB detectors interfaced.
- No more commercial Vendors

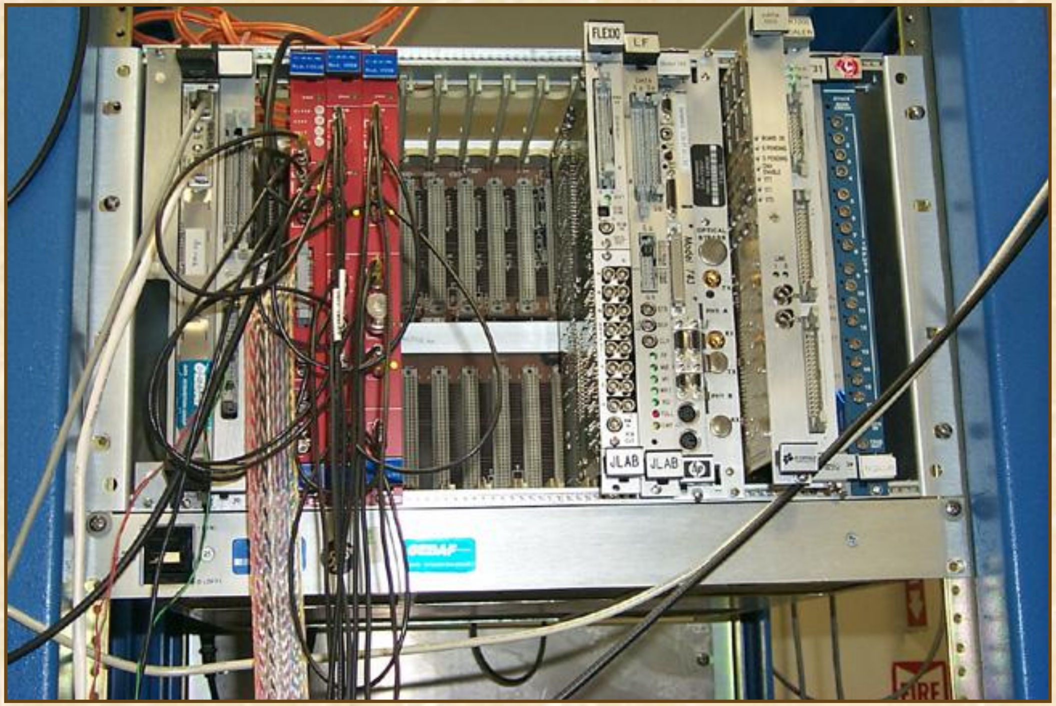


Full Crate



VME (Versa Module Europa)

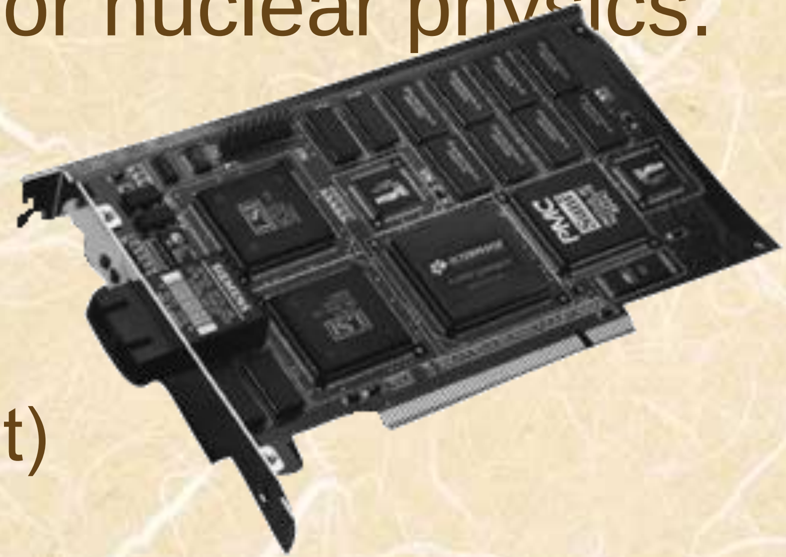
- International standard for interconnecting modules.
- 32/64 bit bus (80MB/s)
- Large number of commercial products (used heavily in the military).
- VME64X provide bandwidth options (160-320 MB/s).
- Currently transitioning from FASTBUS



SFI FASTBUS to VME Controller

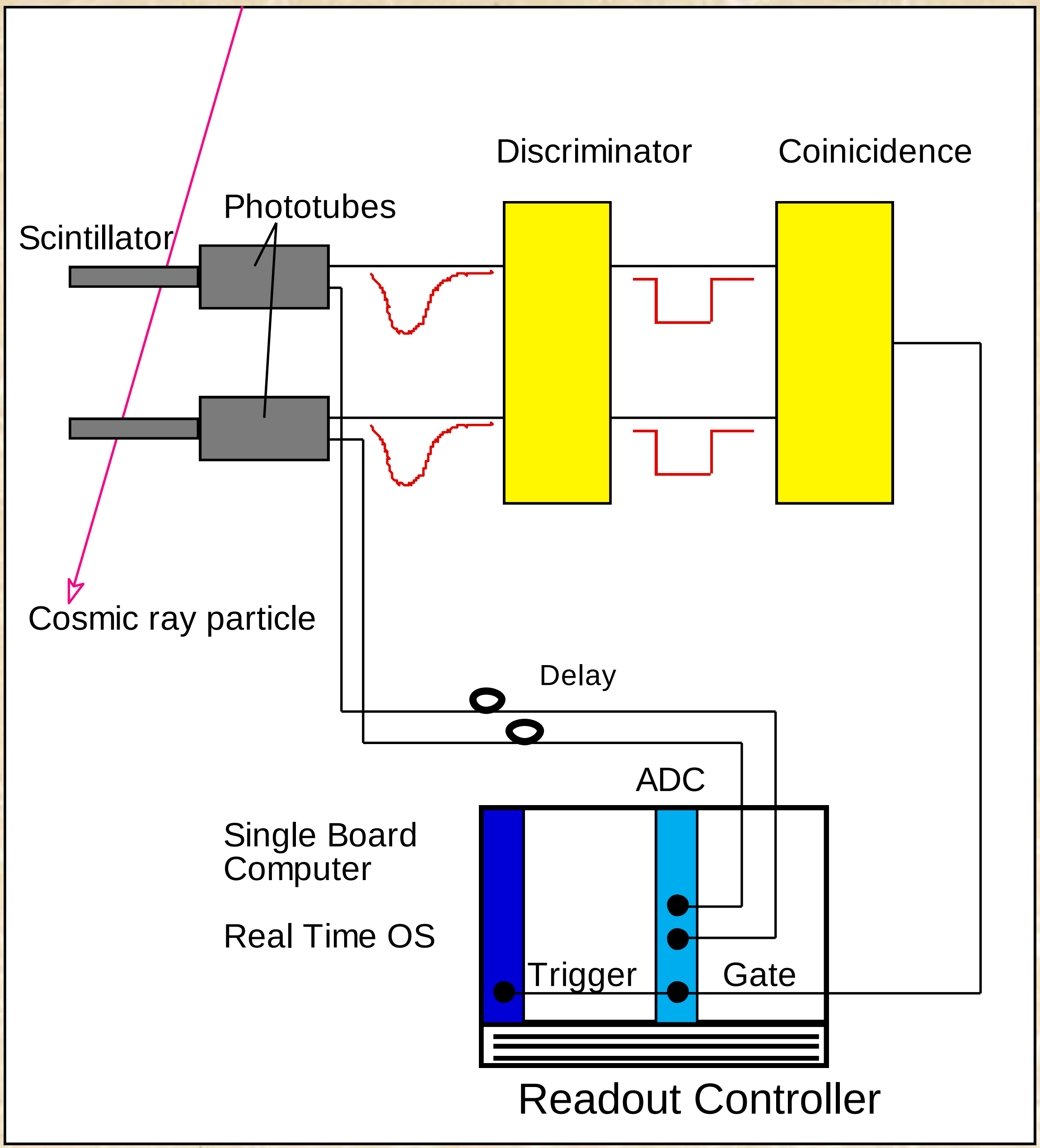
PCI (Peripheral Component Interconnect)

- It's in every PC you buy today.
- Fast 64 bit bus (33-66MHz, 266 - 532 MB/s).
- Not much specialization for nuclear physics.
- Small board size.
- Different "flavors"
  - PCI-X (133 MHz)
  - cPCI (Bus "module" format)
  - PMC (daughtercard)
  - PCI-Express (PCIe serial)

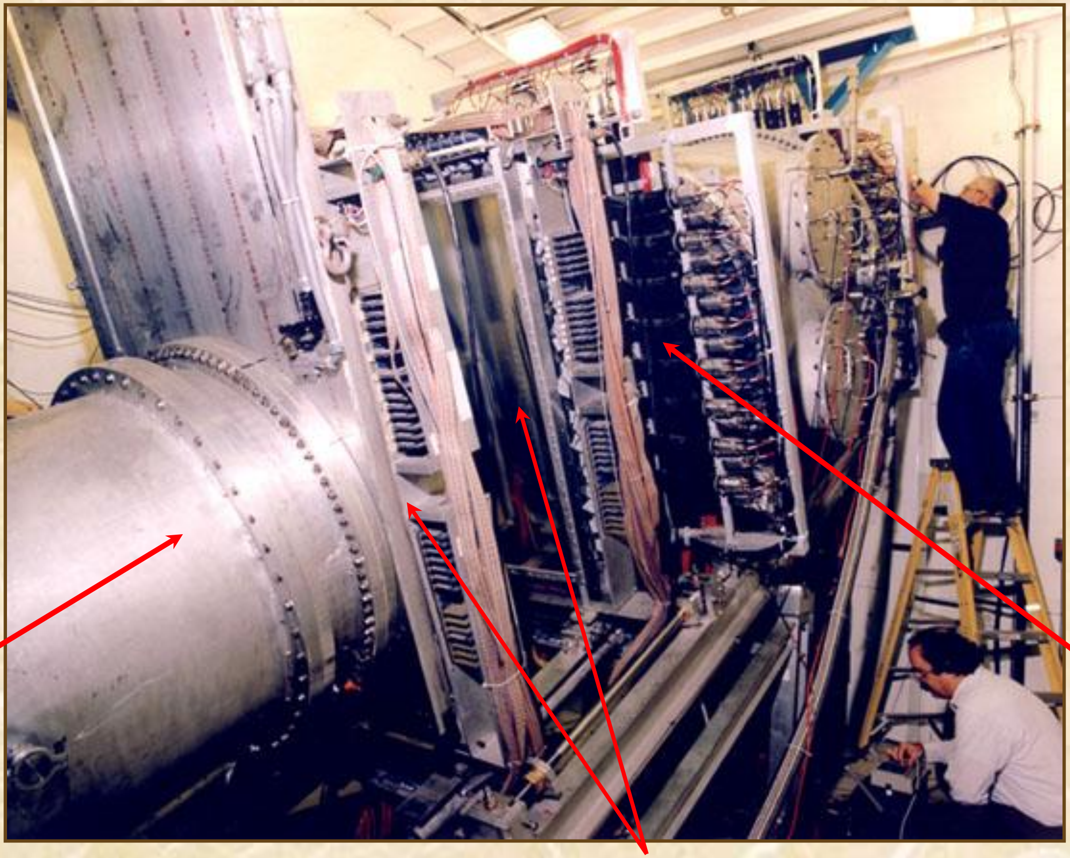


Interface to Detectors

- The Trigger and ADC "*Gate*" begin the conversion and readout phase
- Fast (real-time) response to trigger is important to minimize dead-time



HMS Detectors



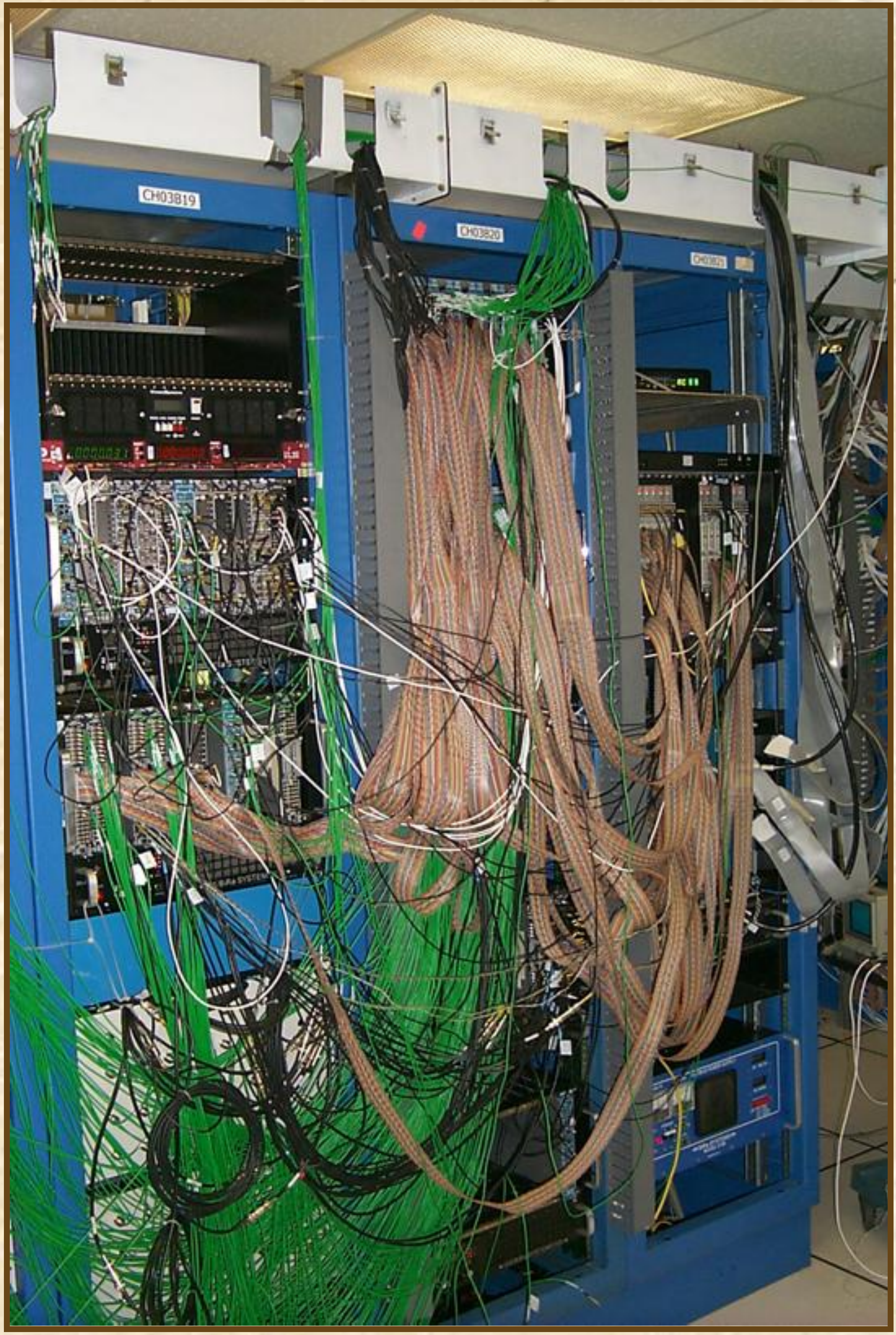
Scattered Beam

Drift Chambers

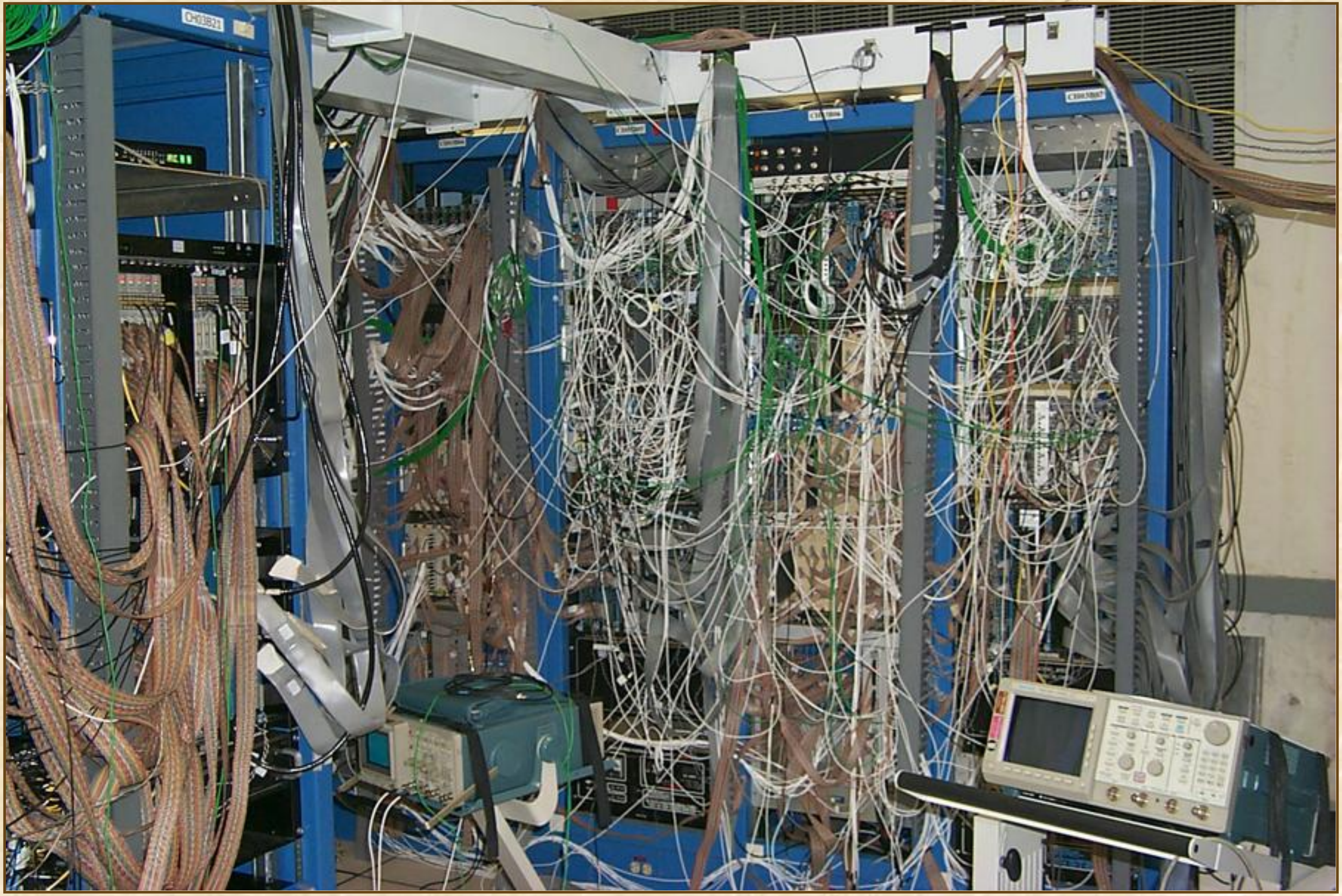
Phototubes

Hall A Spectrometer

The Real World



… and it gets worse!



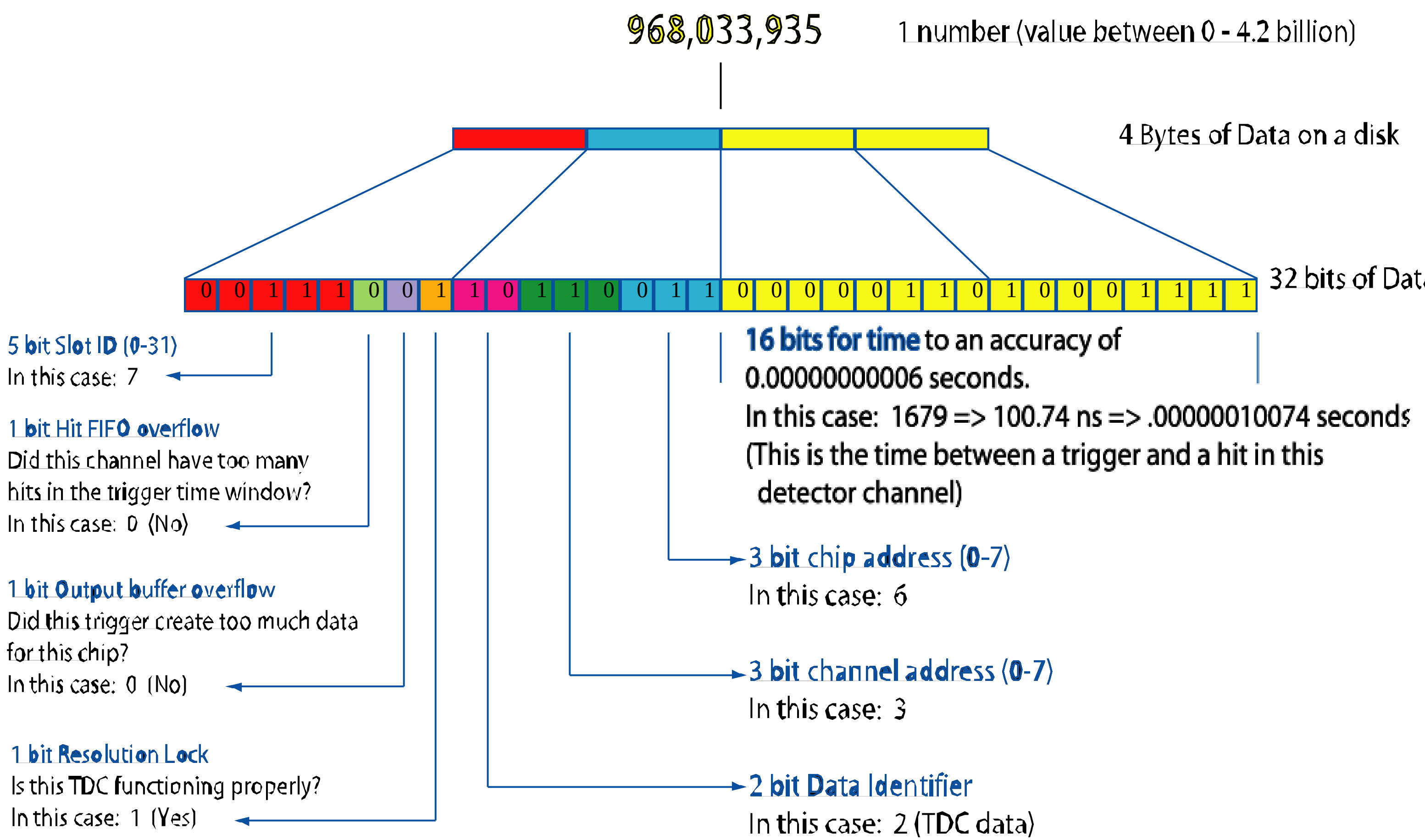
Event formatting

- The data comes from different detectors.
  - Need to identify the detector
  - Need to identify which event this came from
  - Need to make analysis easier.
- There is a lot of data
  - Format must be compact
- Analysis can take years
  - Format should be *self documenting*

What's in a Number?

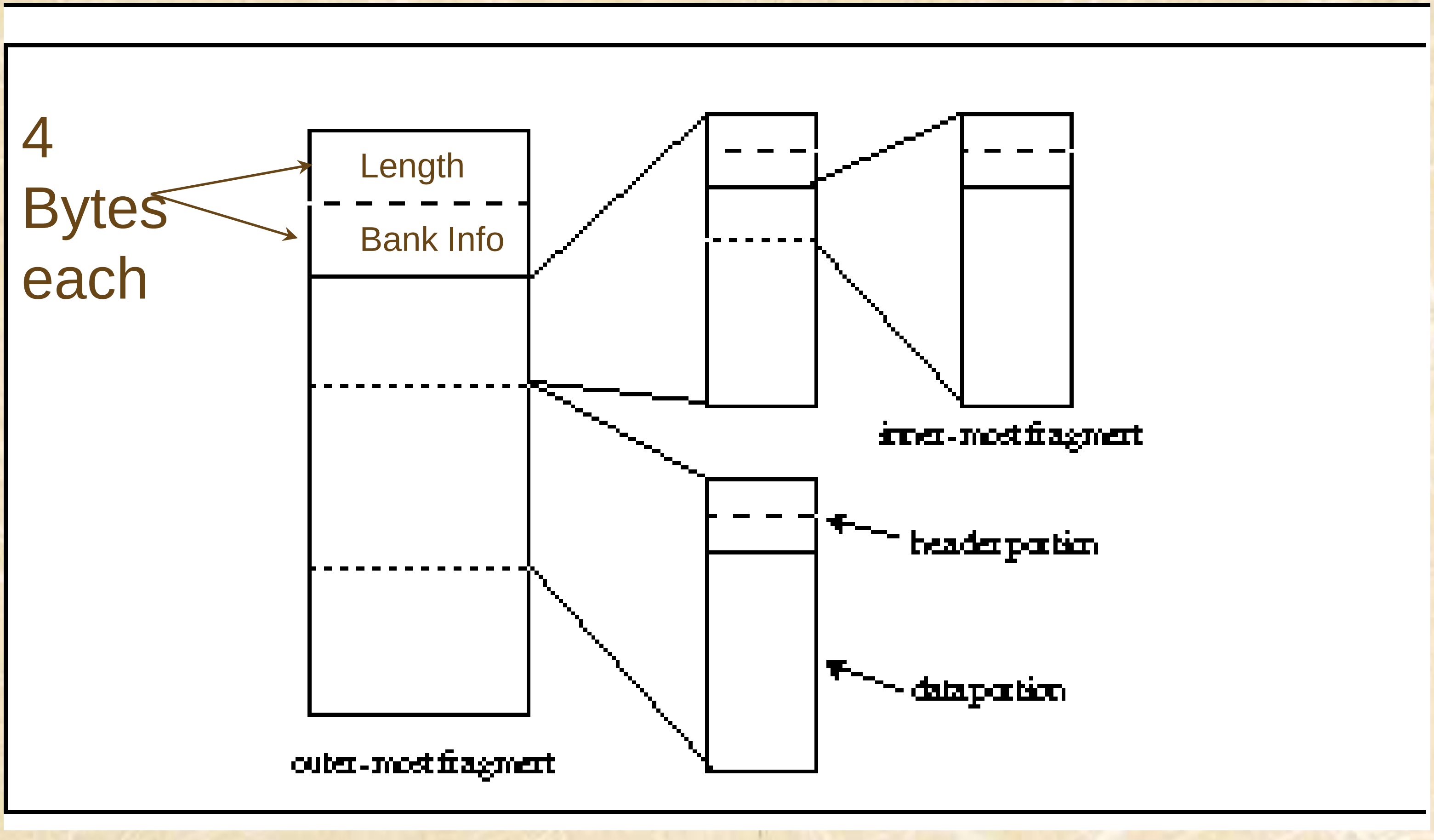
For each detector channel a number is generated when a trigger occurs. There may be many thousands of detector channels for a given physics experiment, and there may be many thousands of triggers generated per second. There is the potential for a HUGE amount of data…

Physicists must be very efficient in the storage of information. So what does typical detector TDC data look like?



Example format

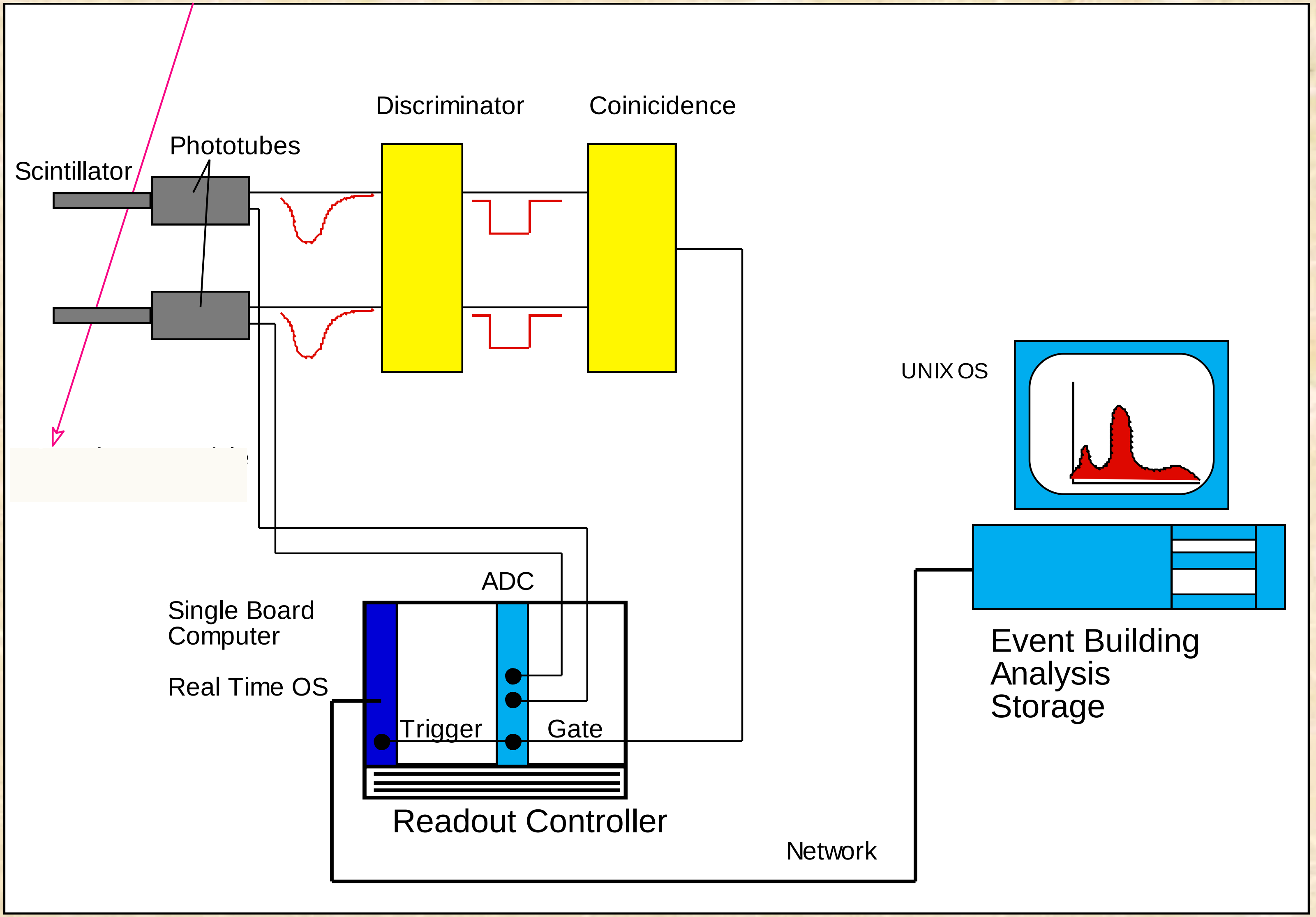
Pack data in “*banks*” and provide “*layering*” of information



Data flow

- Once all the data has been digitized it must be collected into a central place for storage.
- How the data is moved from the detector readout to the storage medium depends on several factors.
  - Available technology
  - Event size and trigger rate
  - Your budget!!
  - Personal taste
- More in DAQ architecture section later.

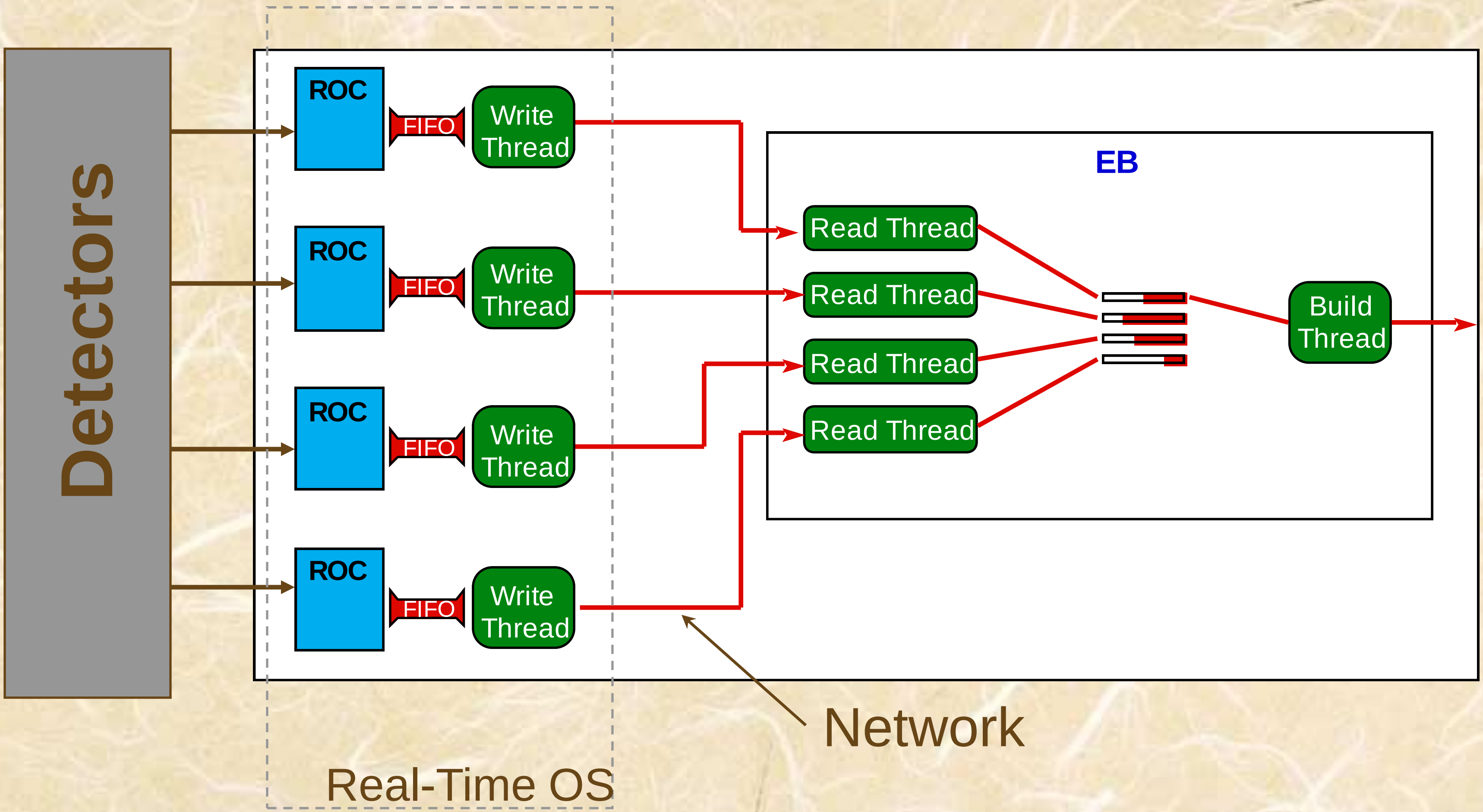
The Physics data flows on…



Event Building

- The detectors are spread over a physical volume of space.
- Bits and pieces of events arrive at different times from different places.
- All the parts of the event need to be collected together and packaged with other information needed by the analysis.
- The *Event builder* is a very fast collating machine.

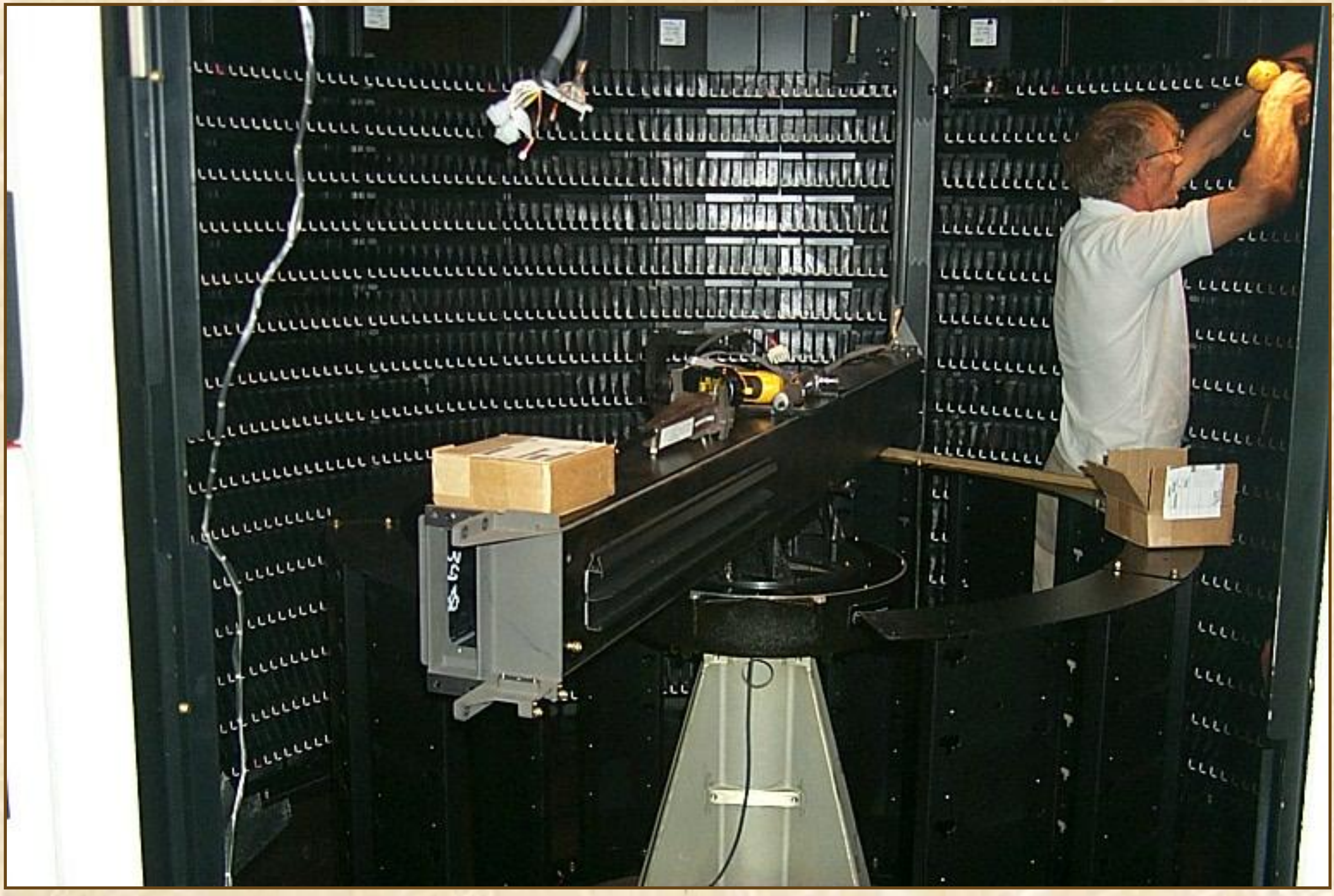
CODA "Push" Architecture



Event storage

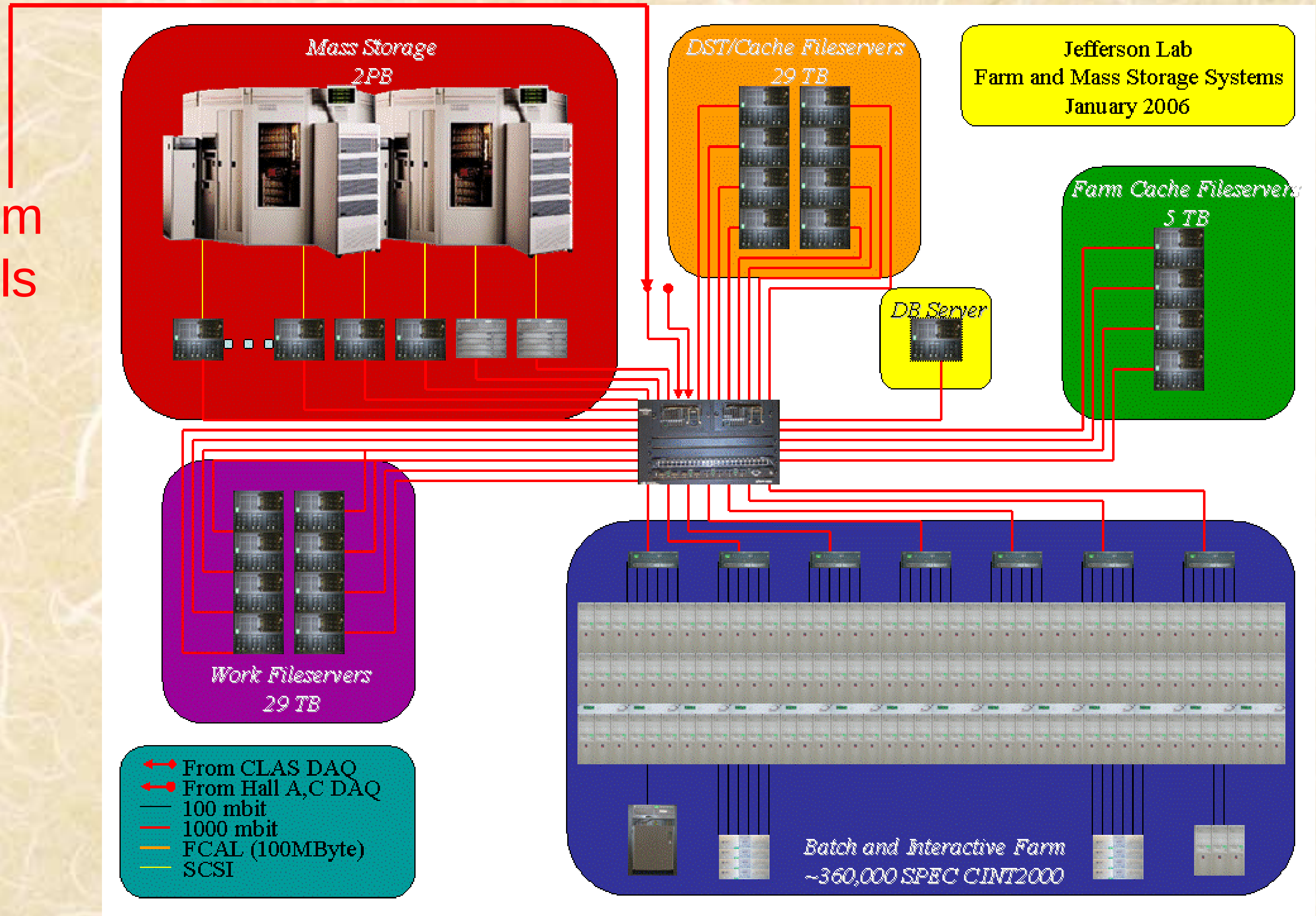
- Since the goal is to store data we need somewhere to put it.
- Physics experiments generate a lot of data. At JLAB 2-35 MB/s per Experiment.
- The fastest method is to Disk!!
- The most cost effective method is tape.
- Stage data to disk -> then backup to tape.
- The tape drives must be fast (and robotic). Fast tape drives are expensive!!
- Aim for tape to be the limiting factor in DAQ speed.

Tape Silos



Storage at JLAB

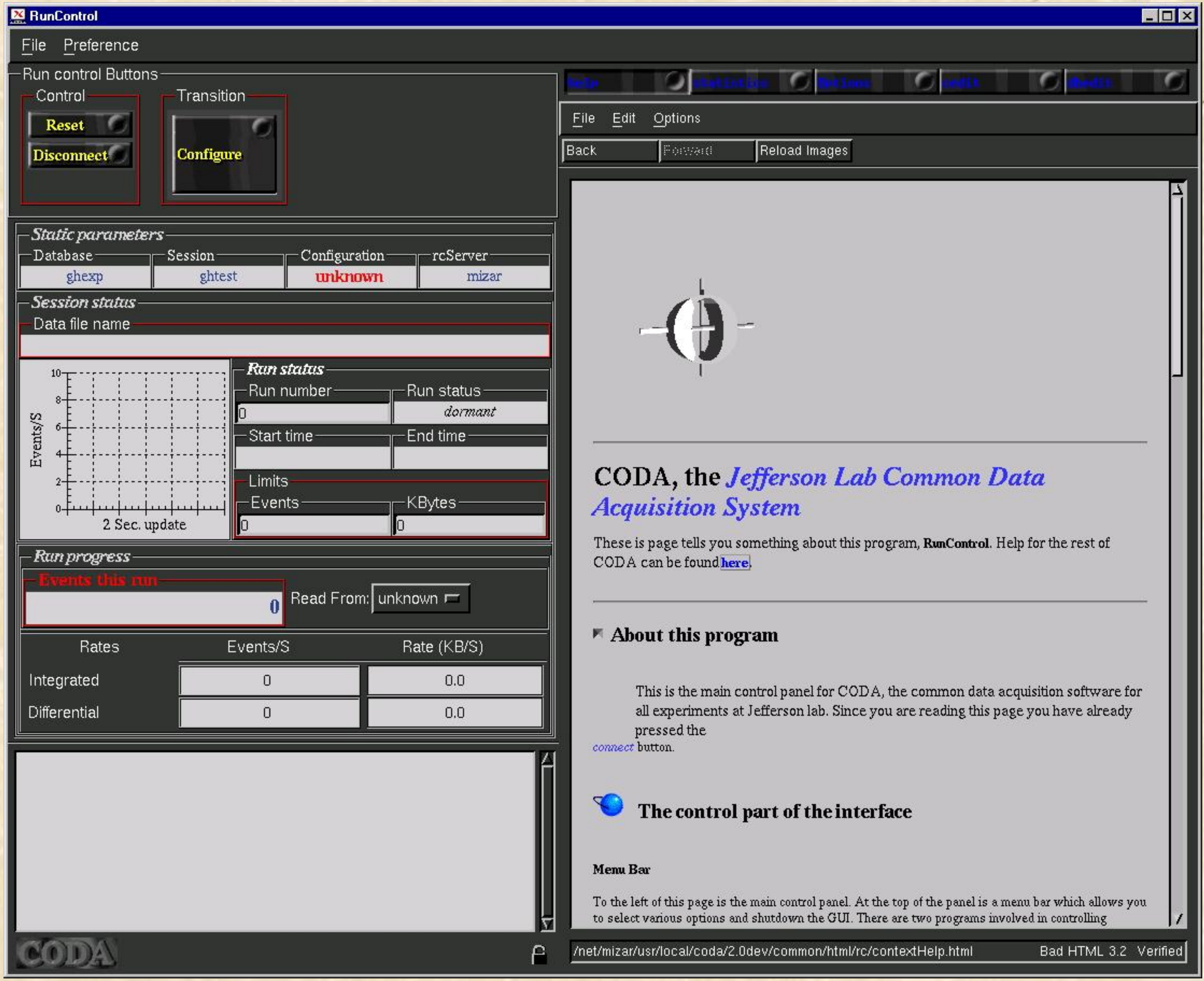
From Halls



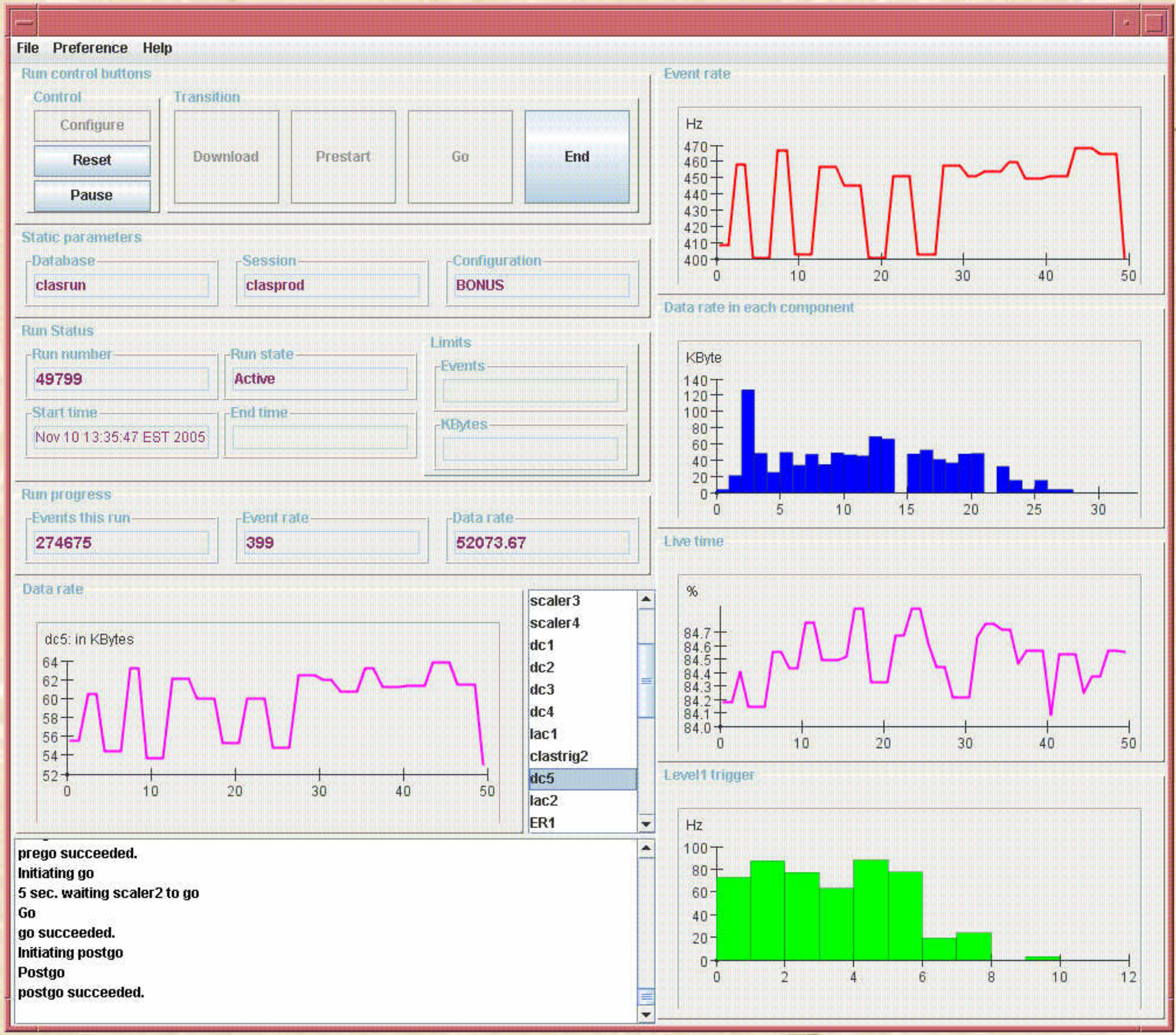
Run Control

- Need to start and stop the DAQ
- Place to input parameters which change from run to run.
- Place to read parameters from.
- Automatic monitor of the health of the DAQ system.
- Something nice for the operator to look at.

CODA X11 Run Control

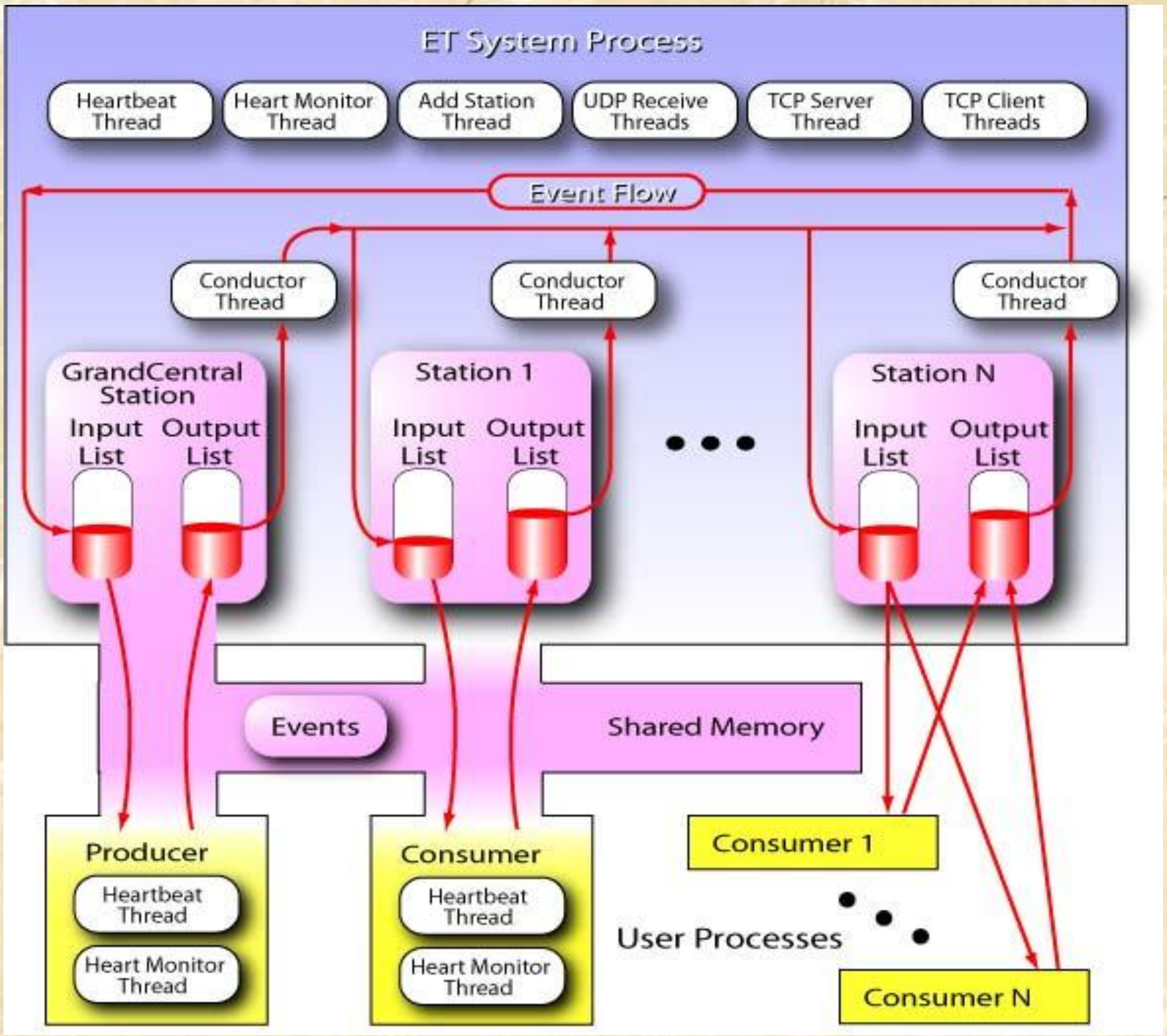


JAVA Run Control



Monitoring/Analysis

- Need to monitor the data quality as it is read.
- Interface between code written by Physicists and code written by DAQ experts.
- Primary goal is to distribute data to anyone who needs it.
- Monitoring must not introduce dead-time.
- CODA Event Transfer “ET” System


ET System Process
Heartbeat Thread
Heart Monitor Thread
Add Station Thread
UDP Receive Threads
TCP Server Thread
TCP Client Threads
Event Flow
Conductor Thread
Conductor Thread
Conductor Thread
GrandCentral Station
Input List
Output List
Station 1
Input List
Output List
Station N
Input List
Output List
Events
Shared Memory
Producer
Heartbeat Thread
Heart Monitor Thread
Consumer
Heartbeat Thread
Heart Monitor Thread
Consumer 1
User Processes
Consumer N

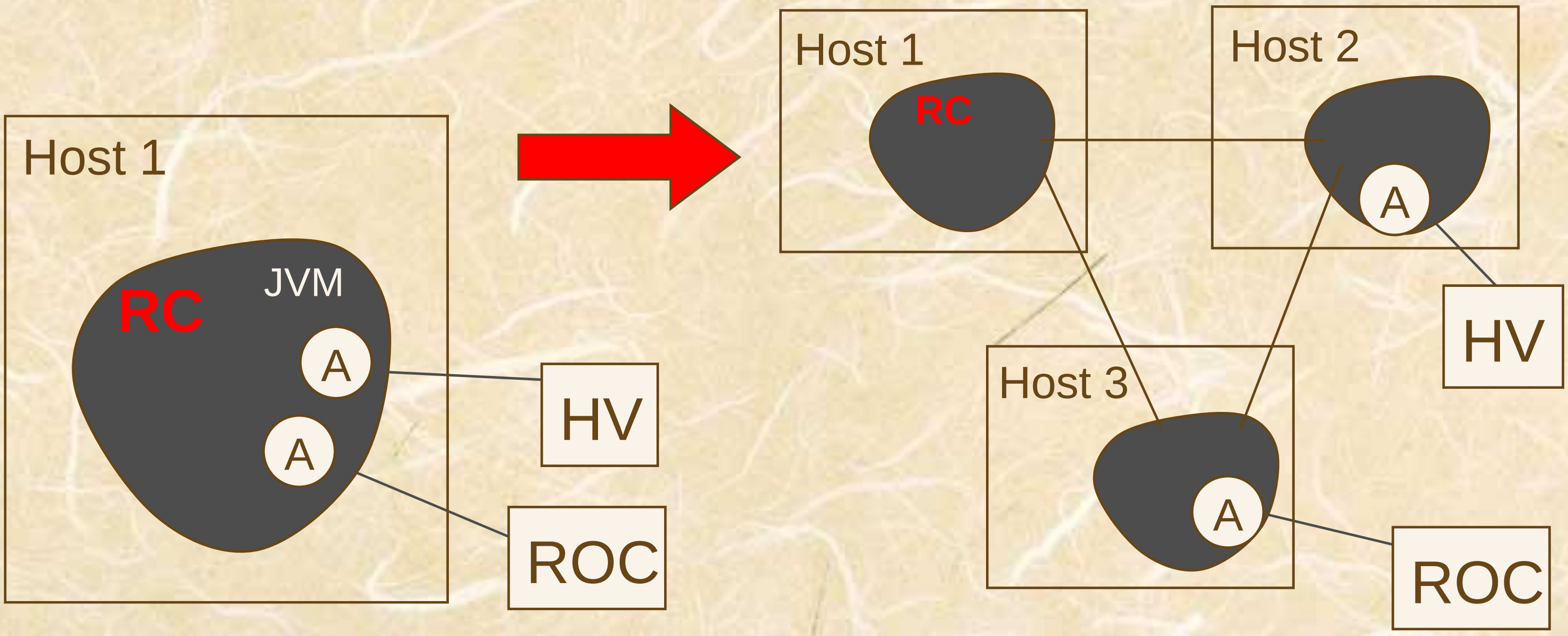
Slow Controls

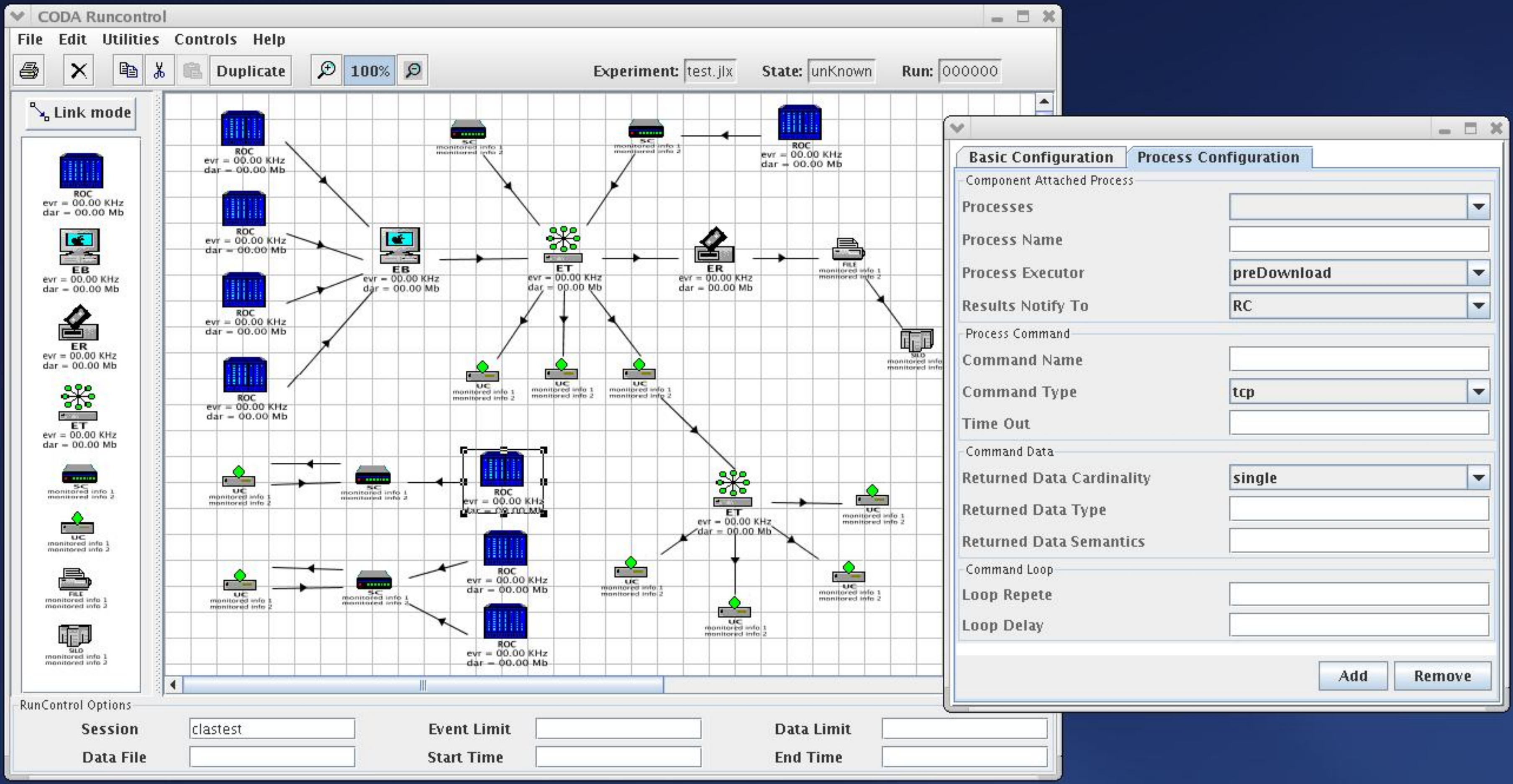
- A topic all by itself.
- Covers all of the other data about the experiment which needs to be acquired.
  - Power supply voltages.
  - Magnetic fields
  - Beam position
  - Target position
  - Vacuum pressure
  - Coffee Maker Status …

The real world, again

Experiment Control

- CODA Java-Based (v 1.5) “Intelligent” agents
- JADE extensions provide a runtime “distributed” JVM.
- Agents provide a customizable intelligence and communication with external processes.





CODA Runcontrol
File Edit Utilities Controls Help
Duplicate
100%
Experiment: test.jlx
State: unKnown
Run: 000000
Link mode
Basic Configuration
Process Configuration
Component Attached Process
Processes
Process Name
Process Executor
preDownload
Results Notify To
RC
Process Command
Command Name
Command Type
tcp
Time Out
Command Data
Returned Data Cardinality
single
Returned Data Type
Returned Data Semantics
Command Loop
Loop Repete
Loop Delay
Add
Remove
RunControl Options
Session
clastest
Event Limit
Data Limit
Data File
Start Time
End Time

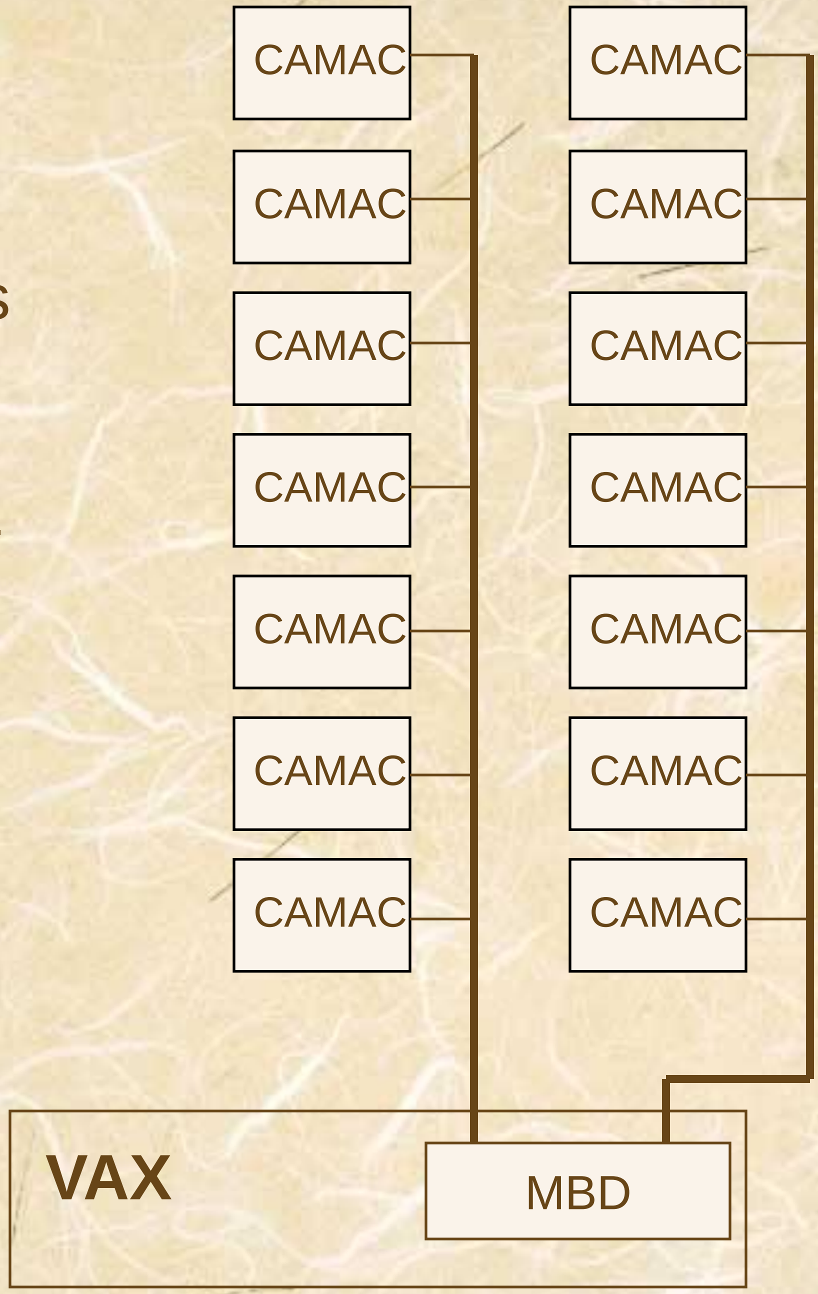
DAQ architectures

- Given all the parts of a DAQ system how are they put together.
- Architecture depends on
  - Event rate
  - Event size
  - Trigger type
  - Available technology

Examples

Older DAQ Systems

- Circa 1980s - DAQ systems were closed and custom-built based on the detectors. -------->
- Single Mainframe CPU systems processed the data.



Break out of a closed system and Rise of the Network

(1980s-90s)

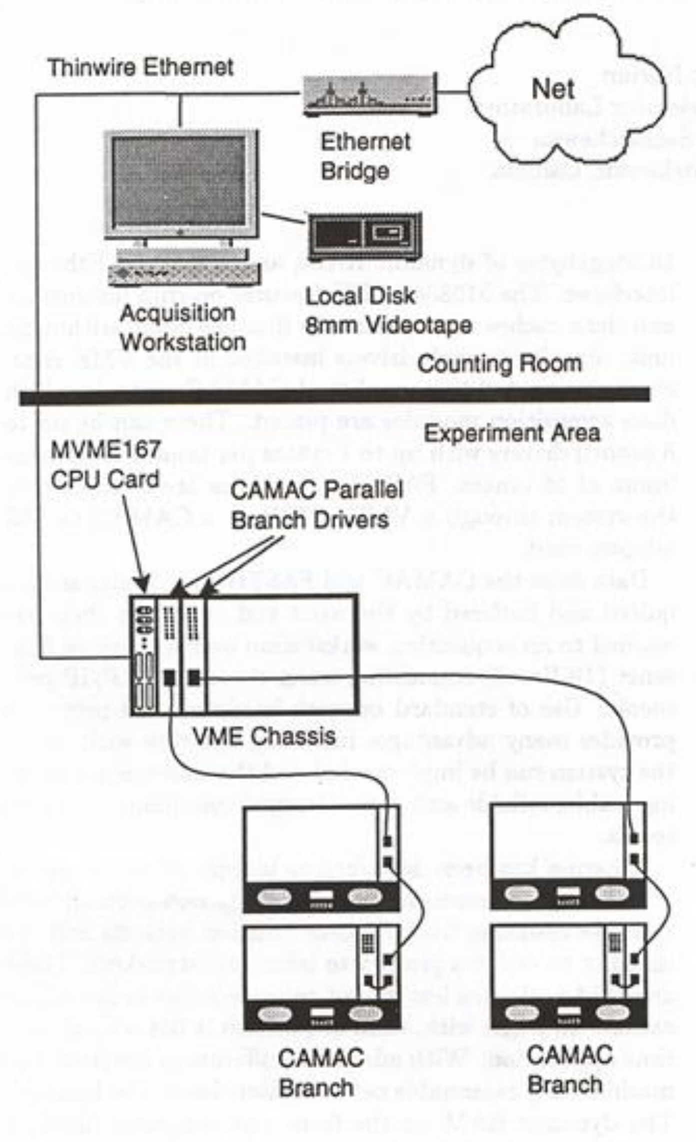
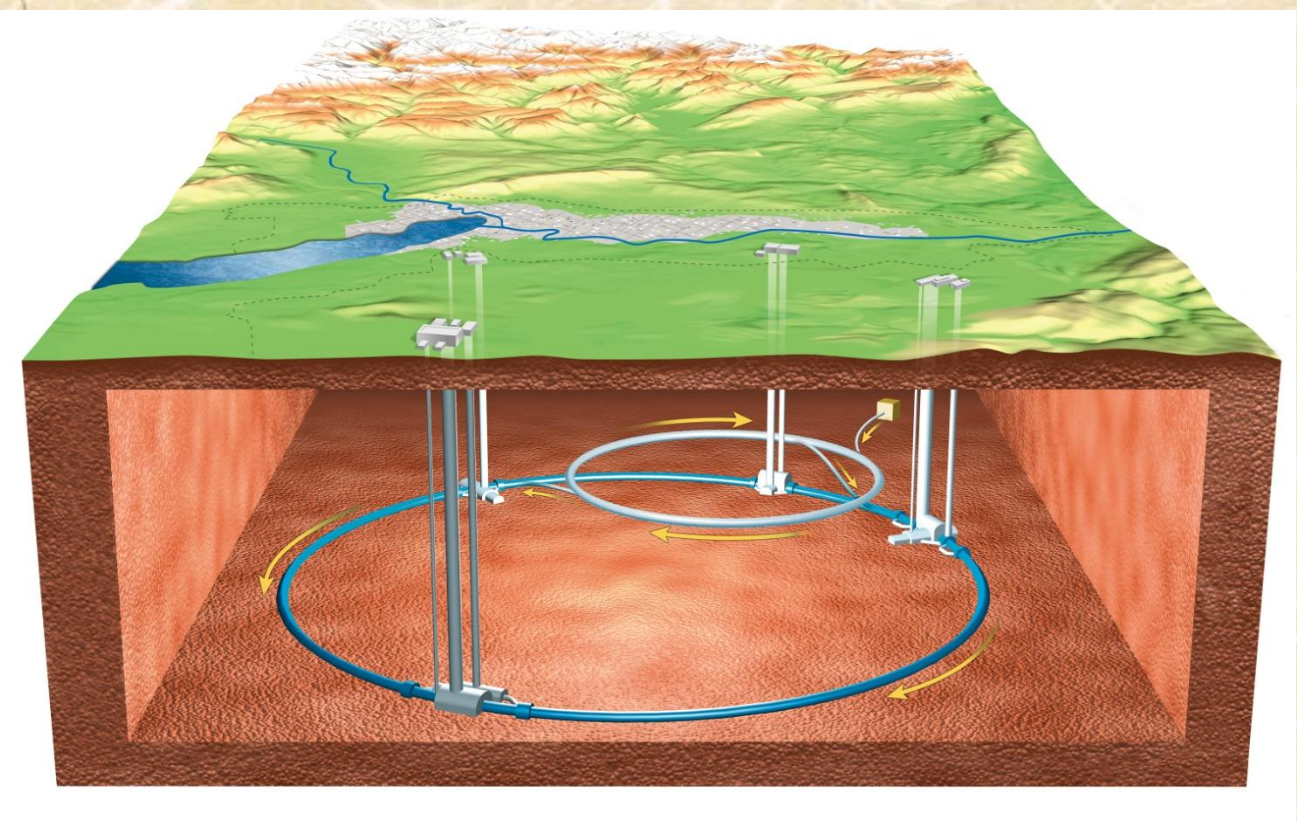


Figure 1: Hardware Configuration

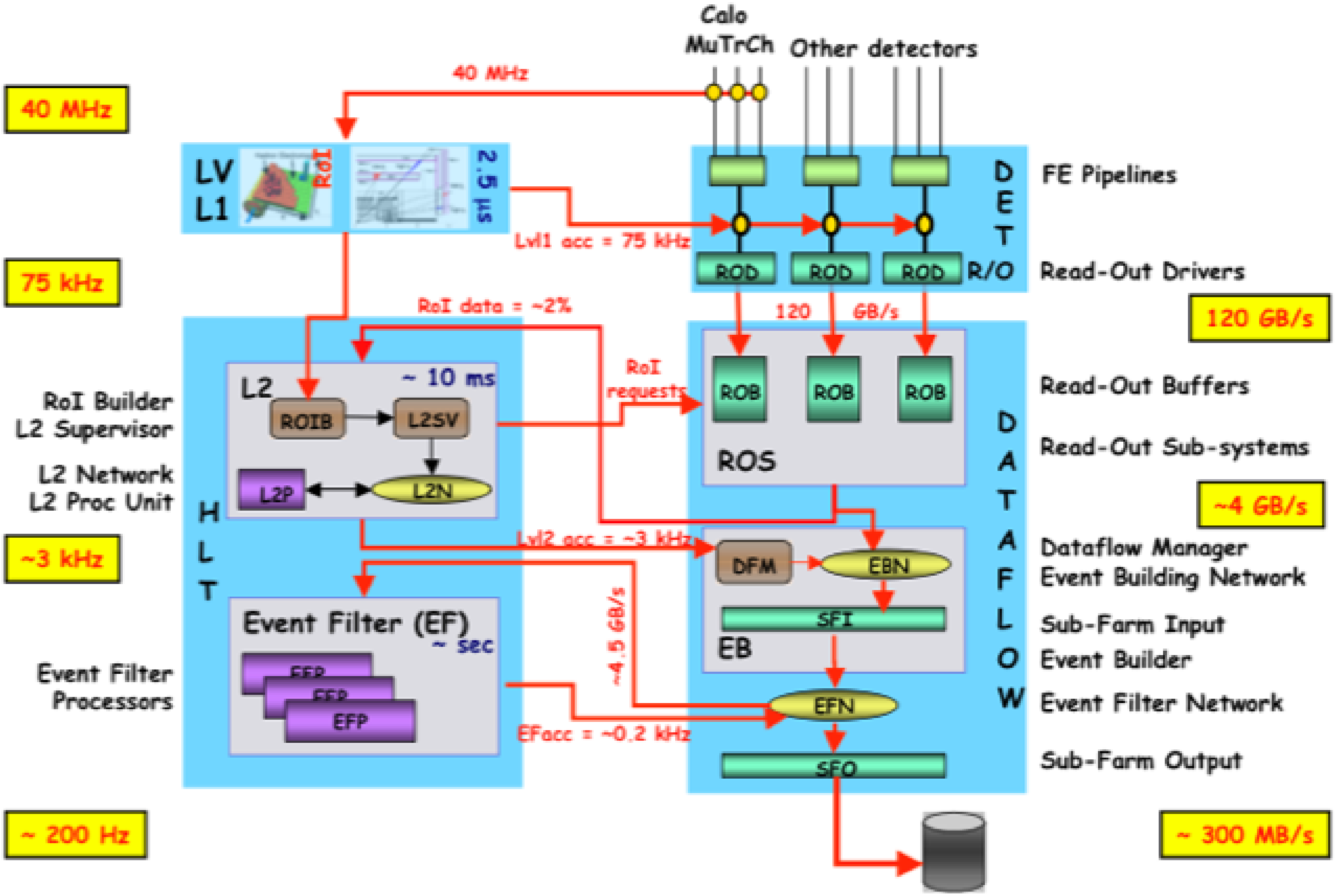
LHC - at CERN (2007->…)

14 TeV Collider
>16 mile circumference
4 primary experiments

- ATLAS
- CMS
- LHCb
- ALICE

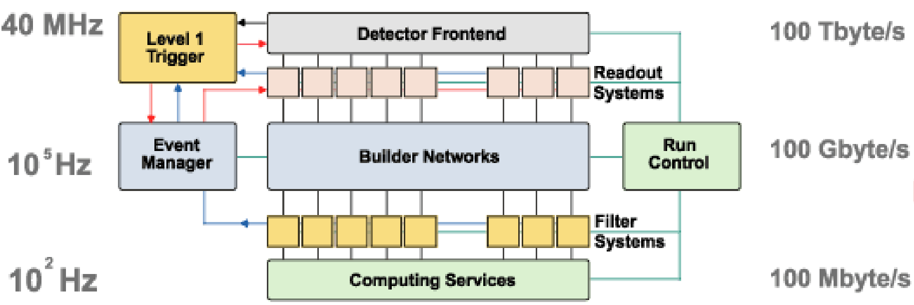


ATLAS Trigger/DAQ system: architecture

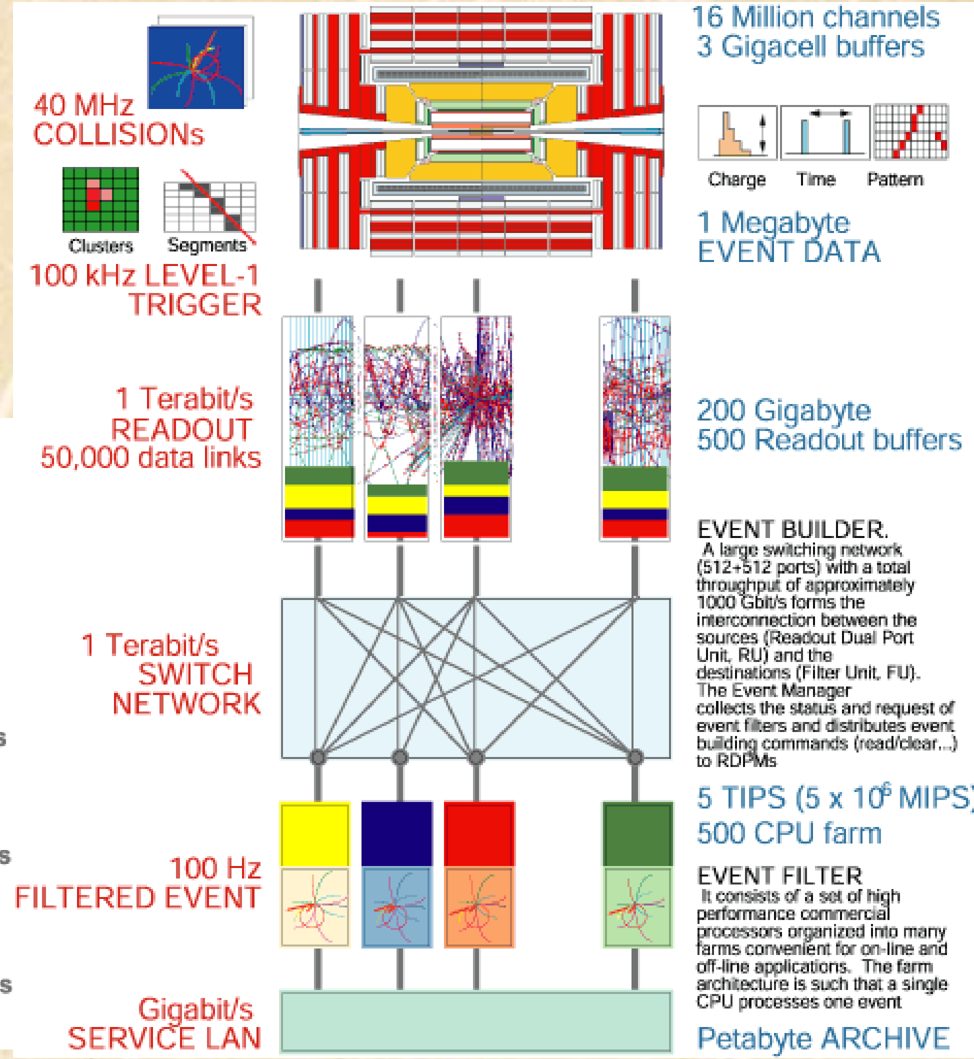


CMS Trigger and DAQ

Data Acquisition Main Parameters	
Collision rate	40 MHz
Level-1 Maximum trigger rate	100 kHz
Average event size	1 Mbyte
No. of electronics boards	10000
No. of readout crates	250
No. of In-Out units (200-5000 byte/event)	1000
Event builder (1000 port switch) bandwidth	1 Terabit/s
Event filter computing power	5 10^6 MIPS
Data production	Tbyte/day



Trigger and Data Acquisition baseline structure

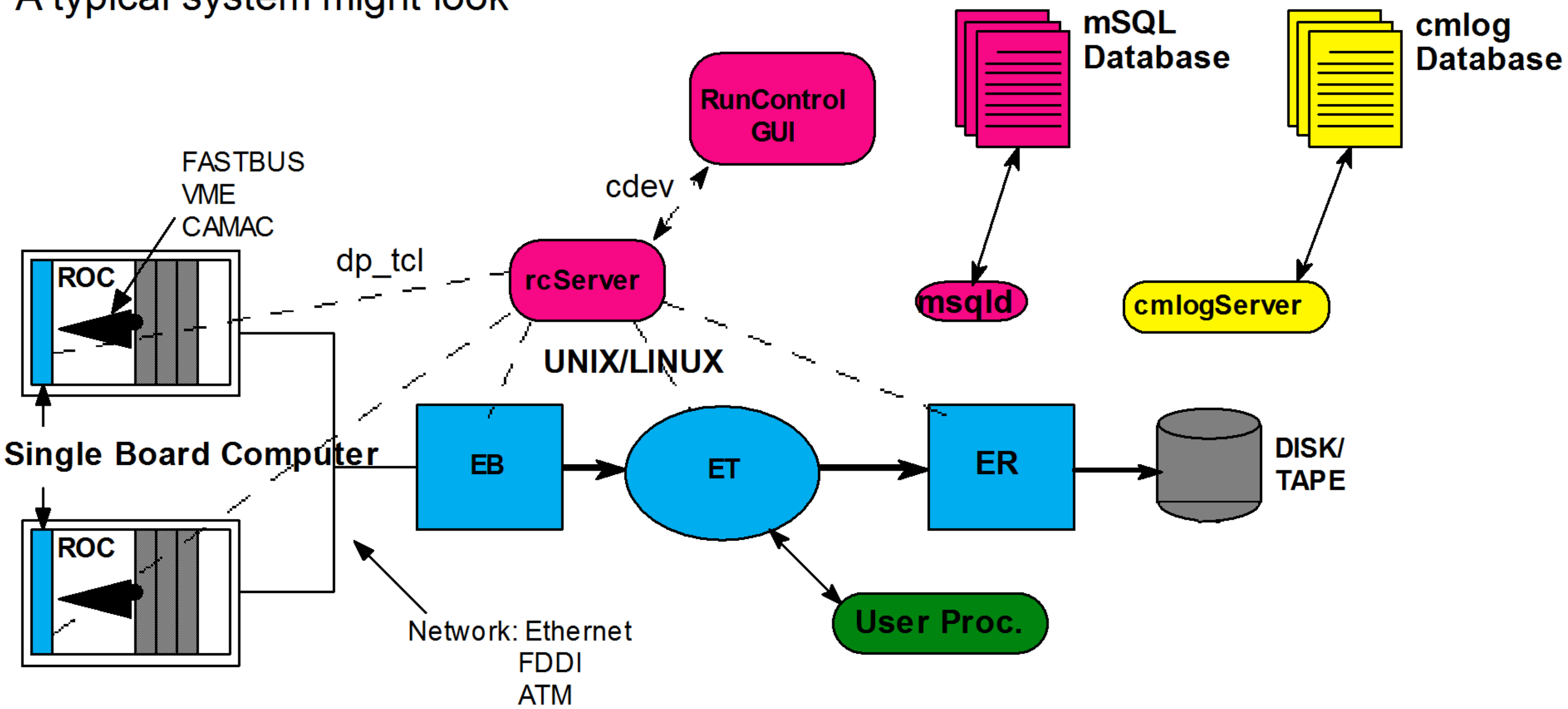


What is this CODA stuff?

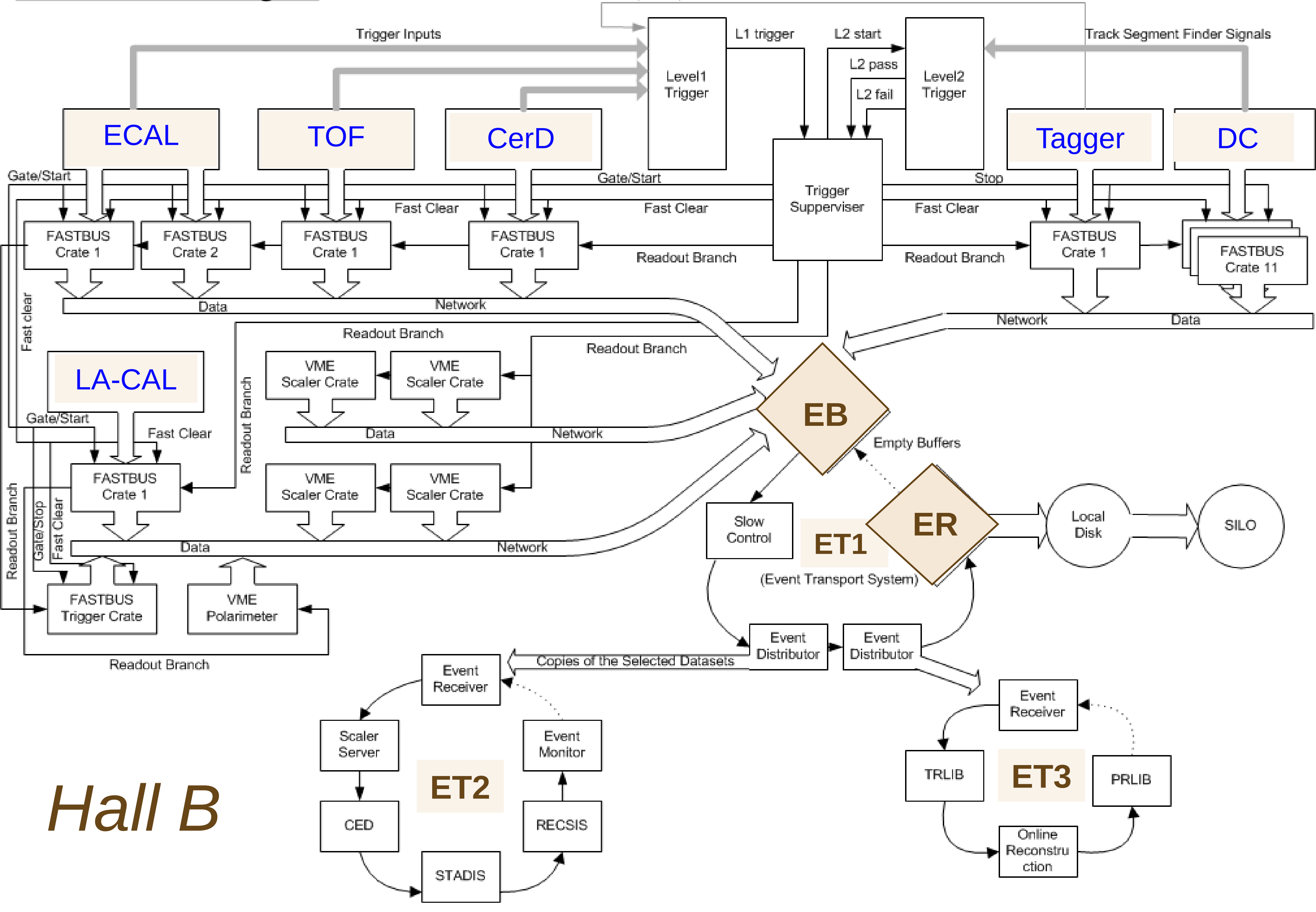
- **C**EBAF **O**nline **D**ata **A**cquisition
- CODA is a software toolkit with some specialized hardware support.
- Modular software components use the *network* for inter-process communication and event transport.
- Use *open standards* and minimize the use of commercial software while maximizing use of commercial hardware.
- DAQ systems for each experimental Hall can be "*built-up*" from common components to fit their needs.

CODA is a software toolkit from which data acquisition systems with varying degrees of complexity can be built.

A typical system might look



CLAS DAQ Block Diagram



Vardan Gyurjyan
11.1.2002

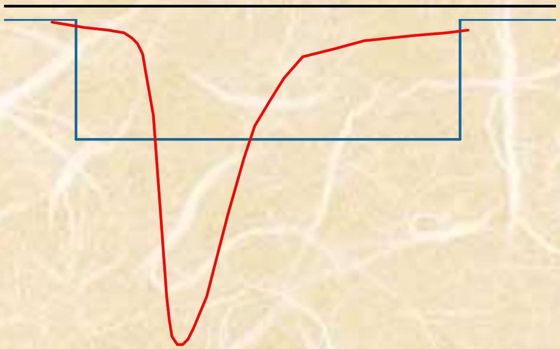
What next?

- Interesting Physics becomes more experimentally difficult (“good” events are more rare)
- Current DAQ is reaching “Limits” of performance
- Technology is always changing.
  - FASTBUS is DEAD as a standard.
  - Computer hardware becoming faster (CPU, RAM, NET)
  - More can be done in software (Real time moves to HW)
  - Busses have reached limits - Hi-speed(2.5-10 Ghz) Serial/Fiber.

    (PCI-X 133MHz ~8Gbit/s   16x PCIe (2.5Ghz) -> 40Gbit/s)
- Reduce dependence on operating systems.
  - Ultrix -> HP-UX -> Solaris -> Linux ===> JAVA
- “Customizable” hardware is becoming a viable option (FPGAs , DSPs, ASICs).

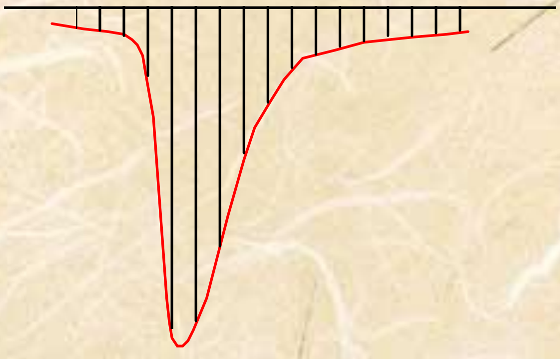
Sampling vs Integration

Traditional "integrating" ADC takes 6-10 µsec to digitize



Generate 1 word representing the charge sum during the gate.

250 MHz Flash ADC samples every 4nsec



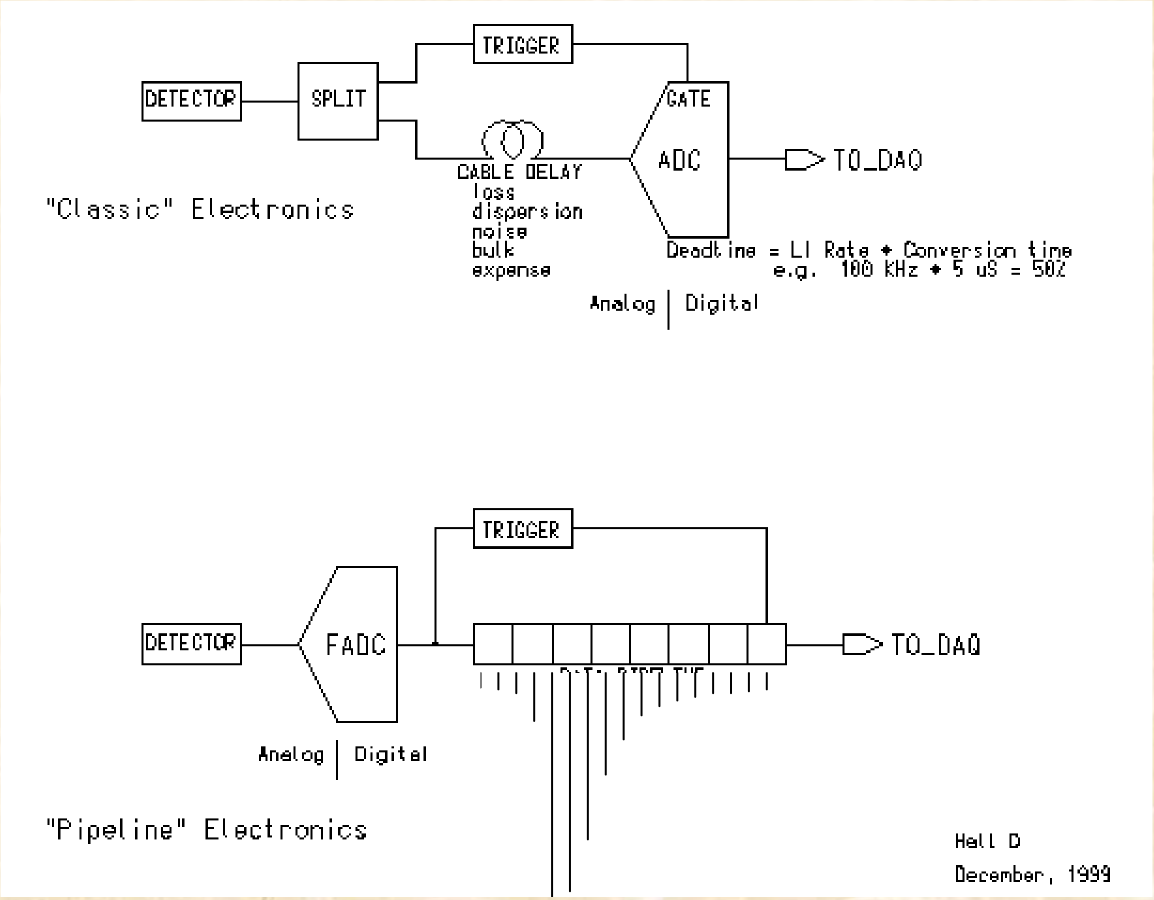
Generates ~10-15 data words that describe the pulse

We can also use these samples to generate the ***Trigger…***

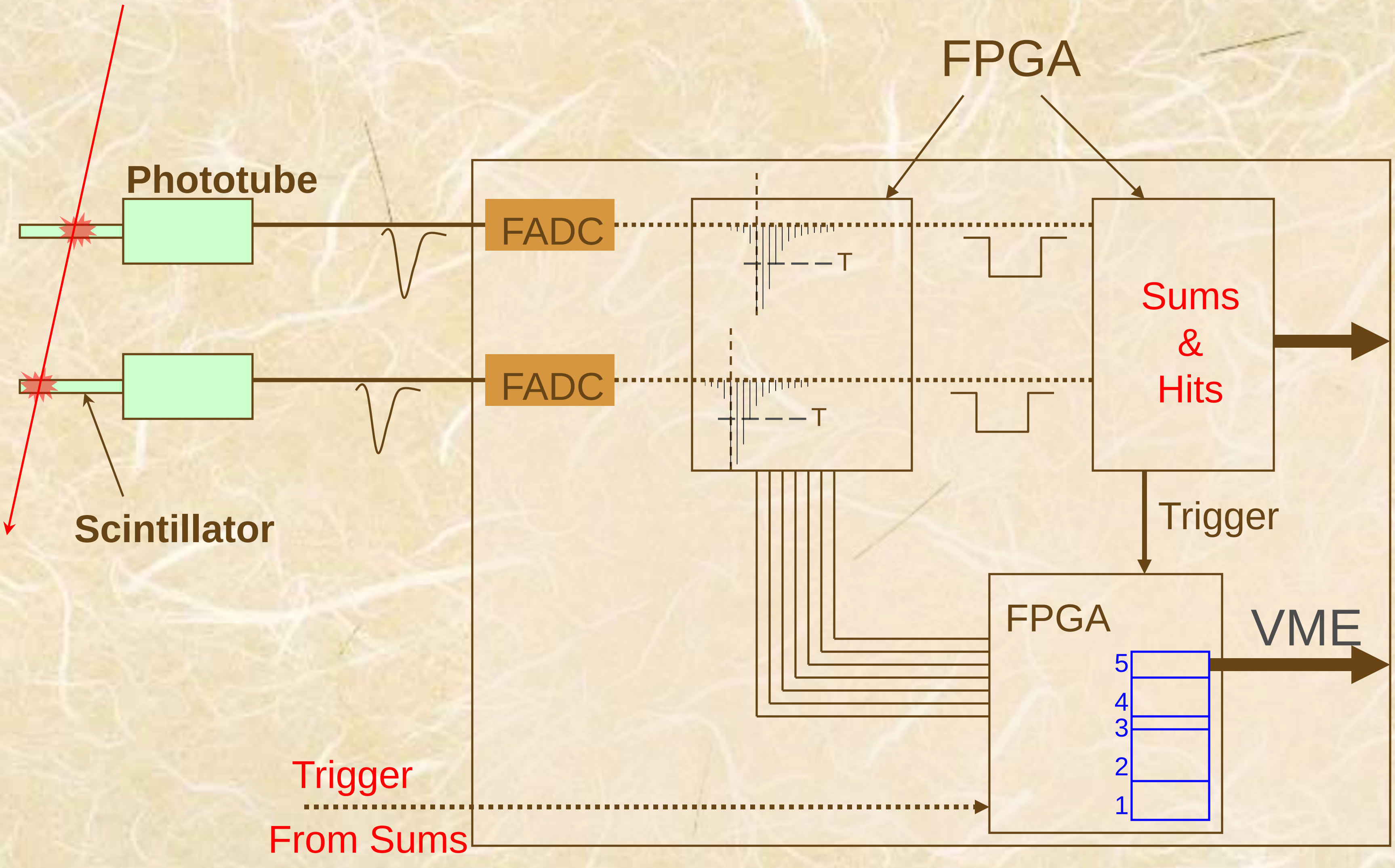
Pipelines (dead-timeless DAQ)

10μs "snapshot" can be stored in memory (~5KB/ADC)

Trigger a look-back and select relevant data.



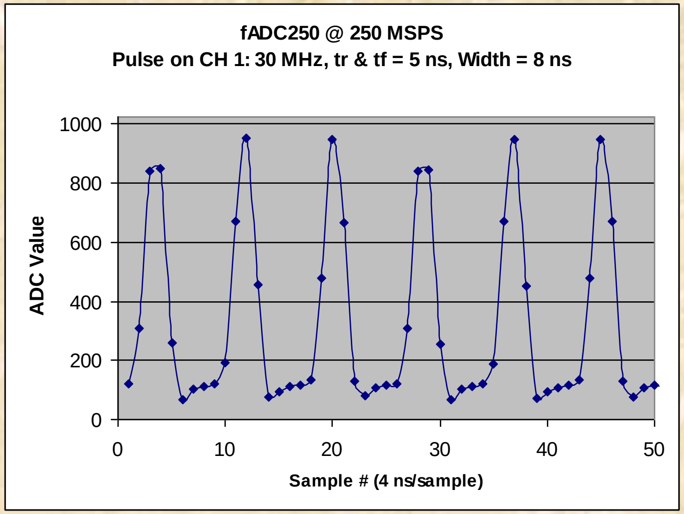
Pipelines (dead-timeless DAQ)

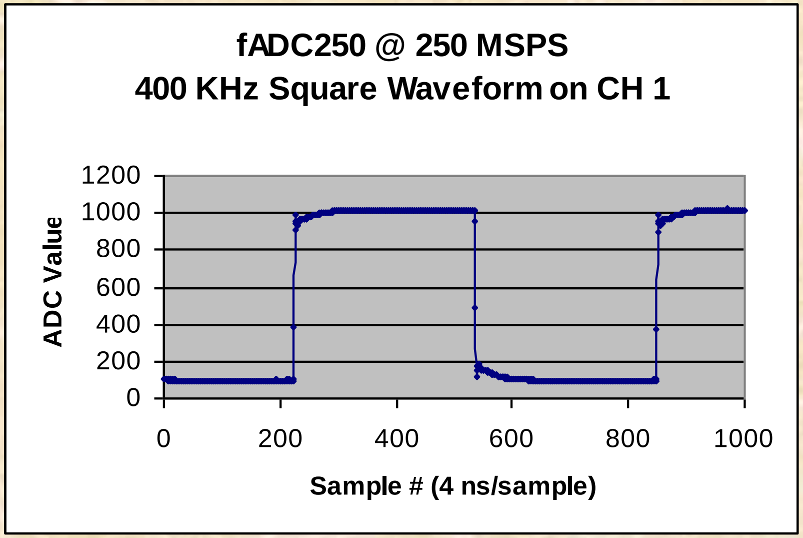


JLAB Flash ADC

FPGAs

250MHz FADC ASICs



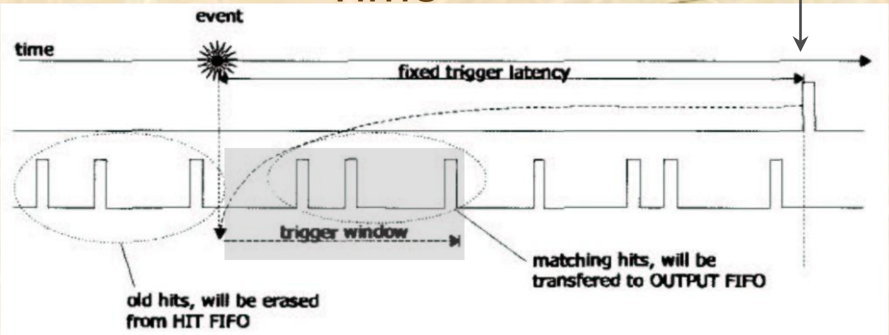


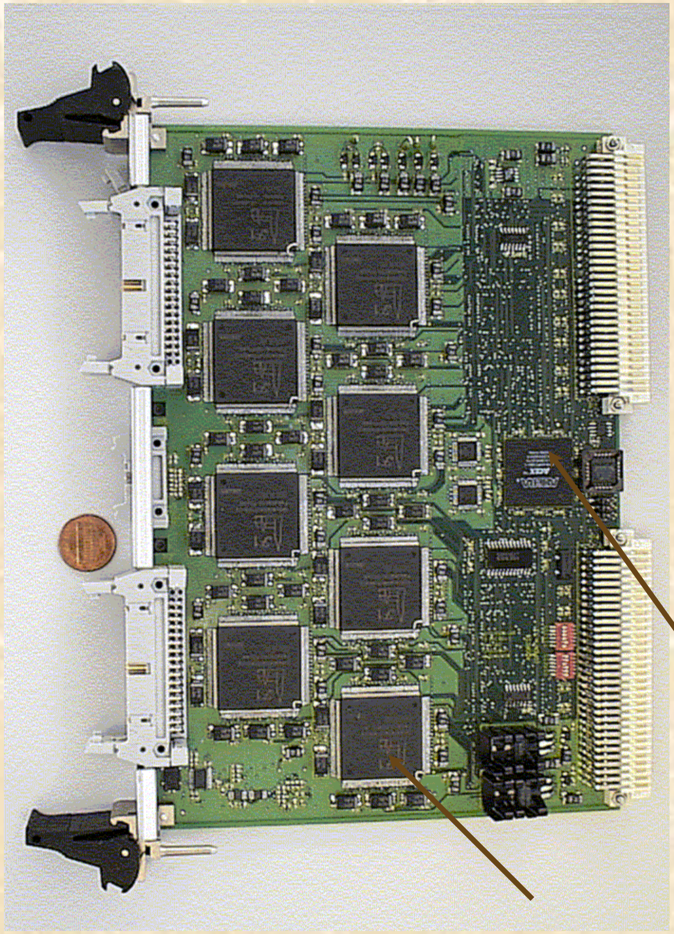
JLAB Pipeline TDC

Time -->

Trigger

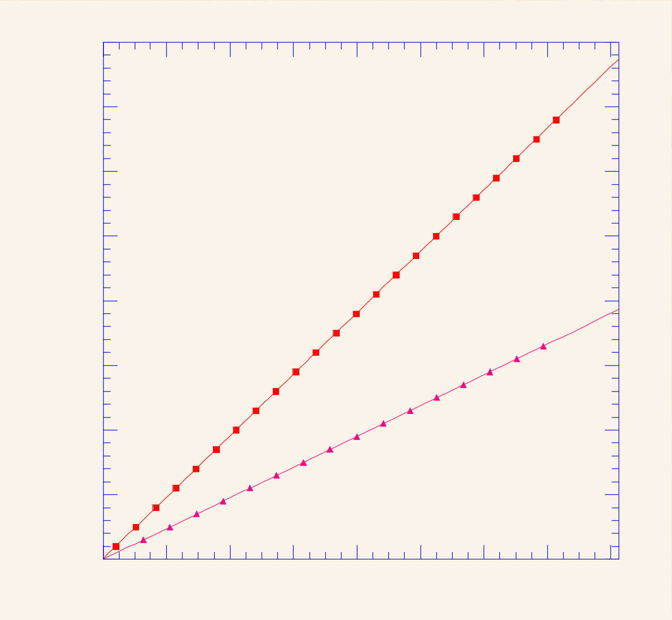
Hits





FPGA

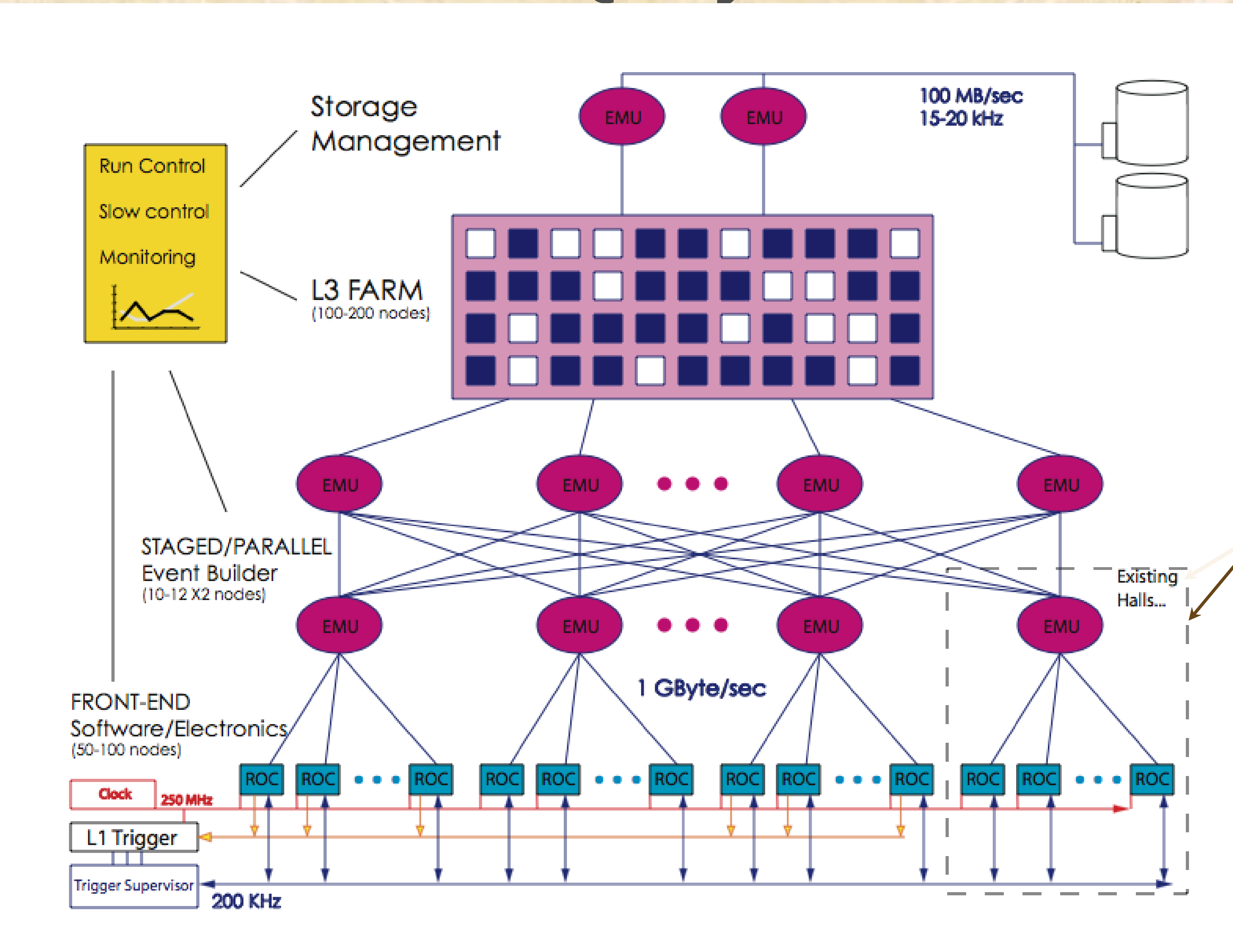
TDC ASIC (8 channels)



CODA3 - Requirements/Goals

- Pipelined Electronics (FADC, TDC)
  - Dead-timeless system
  - Replacement for obsolete electronics
  - Eliminate huge numbers of delay cables
- Integrated L1/L2 Trigger and Trigger Distribution System
  - Support up to 200 KHz L1 Trigger (5µs)
  - Use FADC for L1 trigger input
  - Support 100+ crates
- Parallel/ Staged Event Building
  - Handle 100s of input data streams
  - Scalable (>1 GByte/s) aggregate data throughput
- L3 Online Farm
  - Online (up to x10) reduction in data to permanent storage
- Storage Management
  - Ordering/sorting of built events (at 15-20 kHz, 100 MB/s) to disk

CODA 3 DAQ System



Existing Halls

Thank you!

that’s all folks…