Event building

Il sistema di acquisizione (DAQ) raccoglie dati da *buffers* a diversi livelli (*front end*, intermedi etc.) per metter insieme un "evento" da scrivere su un supporto permanente.

Questa azione si può scomporre in vari passi, non tutti presenti in ogni sistema:

- (*sub-event bulding*)
- *event building*
- (operazioni sui dati)
- (trasmissione a distanza)
- scrittura su supporto permanente

Event building è la procedura per accedere ai dati il più rapidamente possibile, assicurandosi che l'evento è coerente e completo.

In un sistema semplice, che contiene solo un evento alla volta, del quale tutti i dati sono disponibili a partire dallo stesso istante in posizioni predeterminate, *event building* si riduce ad una lettura ordinata dei dati.

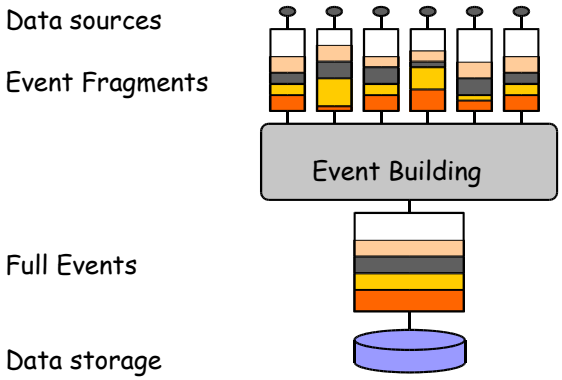
In un sistema complesso:

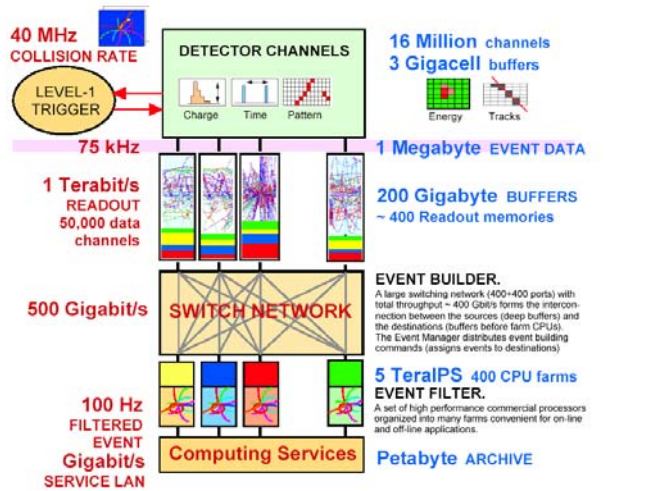
- I diversi rivelatori hanno risposte temporali diverse e pure sono diversi i tempi di conversione dei segnali.

- Quando la frequenza delle interazioni è molto elevata i vari livelli di trigger non riescono a tenere il passo coi dati che entrano: i dati sono *pipelined,* con un puntatore che indica a quale evento appartenevano.

-Alcuni dati sono processati dai vari livelli di trigger e l'evento può contenere tanto i dati come sono stati prodotti quanto una frazione di dati processati. Questi ultimi si trovano in buffers diversi da quelli dei dati d'origine.

Event Building

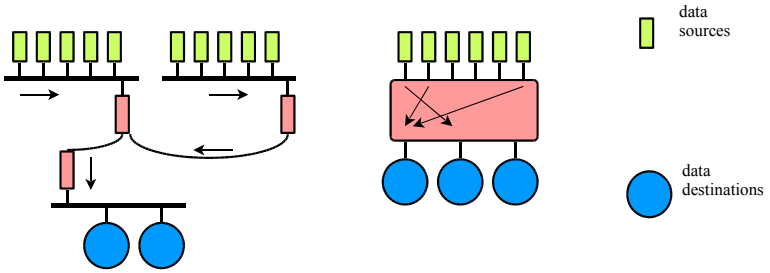




Oltre alla procedura è necessario anche il materiale:

Tradizionalmente, nei sistemi basati su un *bus*, i *buffers* sono montati in *crates*, che contengono controllori di lettura. I *crates* sono interconnessi tra di loro e con i processori destinatari dei dati

Negli esperimenti LHC per raggiungere la banda passante richiesta si usano *switches*



- Tecnologie industriali di commutazione
  - Telecomunicazioni e reti di calcolatori
    - ATM (fino a 9.6 Gb/s)
      - *cells* da 53 bytes, caro
    - Fiber Channel (1 Gb/s)
      - *connection oriented*
    - SCI - Scalable Coherent Interface (4 Gb/s)
      - *Memory mapped* I/O, messaggi da 64 bytes
    - Myrinet (2.5 Gb/s)
      - dimensine del messaggio illimitata
      - *switches* poco cari
      - lo *switch* non ha *buffers*
    - Gigabit Ethernet (fino a 10 Gb/s)
      - poco caro
    - Infiniband (1x, 4x, 12x e 32x 2.5 Gb/s)
      - tecnologia del futuro?

Table 6-1 Network technologies comparison.

CMS DAQ TDR, CERN/LHCC 2002-26, p. 69

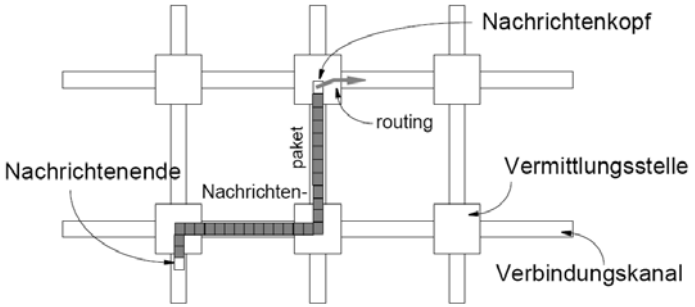
	Switched Ethernet	Myrinet
focus	Local Area Network	cluster I/O
bandwidth	100 Mbps (Fast Ethernet) 1 Gbps (Gigabit Ethernet) 10 Gbps emerging	2.5 Gbps (8b/10b encoded[a])
latency (MPI application[b])	100 µs	10 µs
routing method[c]	destination-based	source-based
switching algorithm	store-and-forward (typically)	wormhole
switch type	shared memory (typically)	crossbar (Clos)
flow control	Xon/Xoff pause frames (optional)	link level Xon/Xoff
medium	UTP5 up to 200 m, fiber	copper up to 3 m (SAN), fiber
MTU size	1500 bytes (user payload) jumbo frames (optional)	unlimited
multicast/broadcast	yes	no
processor on NIC	typically not	yes
market	multi-vendor	single vendor

Footnotes:

a. Due to the 8b/10b encoding of the data the 2.5 Gbps baud rate results in a 2 Gbps effective bandwidth.

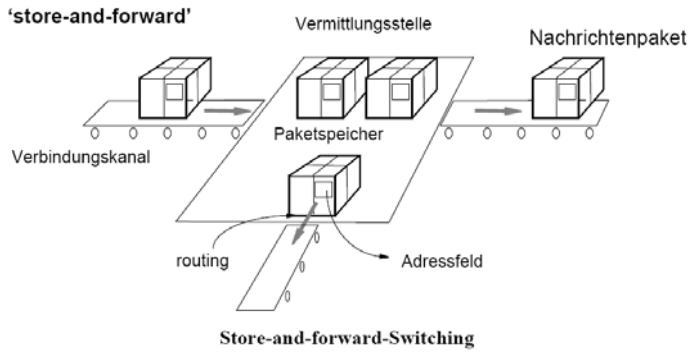
b. MPI (Message Passage Interface) is an industry standard employed widely in high-performance computing.

c. the routing method refers to whether the determination of the path to follow between source and destination is done at the source (source routing) or at each switching node along the way (destination based).

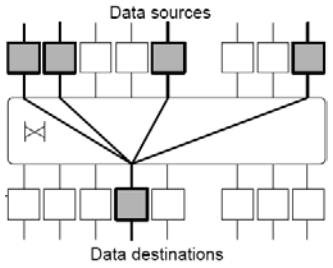


Worm-hole-Strategie

Lehrstuhl für Rechnerarchitektur - Universität Mannheim

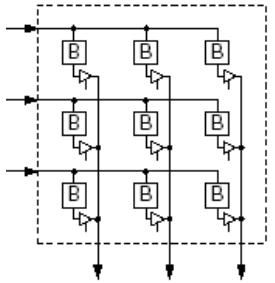


Event Buiding usando uno Switch



Tre difficoltà:
- Impiego efficiente della banda passante di ciascuna connessione
- Bloccaggio in uscita: tutte le sorgenti convergono su una sola destinazione
- Numero di porte molto elevato >128

E' necessario organizzare il traffico (*traffic shaping*) per evitare il bloccaggio



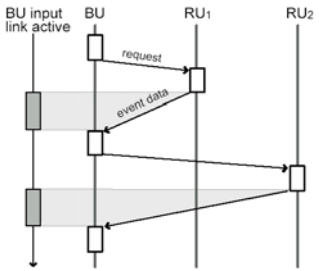
1 - Buffering nello *switch*:

IQ, OQ, CIOQ, VOQ e/o *buffers* interni come in figura >>
Soluzione generalmente cara

2 - Destination driven:

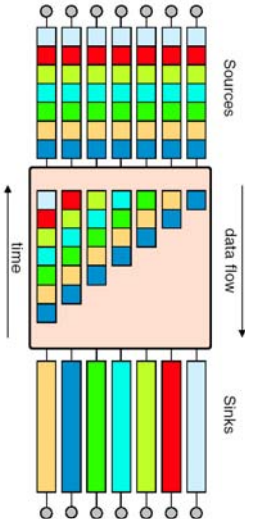
Il destinatario (BU) chiede alla sorgente RU_1 di inviargli l'evento *k*; alla fine della trasmissione chiede alla sorgente RU_2 di inviargli l'evento *k* e, quando ha ricevuto tutto l'evento *k*, chiede alla sorgente RU_1 di inviargli l'evento *k* + 1 ....

I *links* sono poco utilizzati!



3 – Barrel shifter:

A monte dello *switch* si organizza il traffico in un modulo (*sorter*) che prepara i pacchetti per ciascuna destinazione.
Funziona con un *crossbar* o con uno *switch* di Clos.

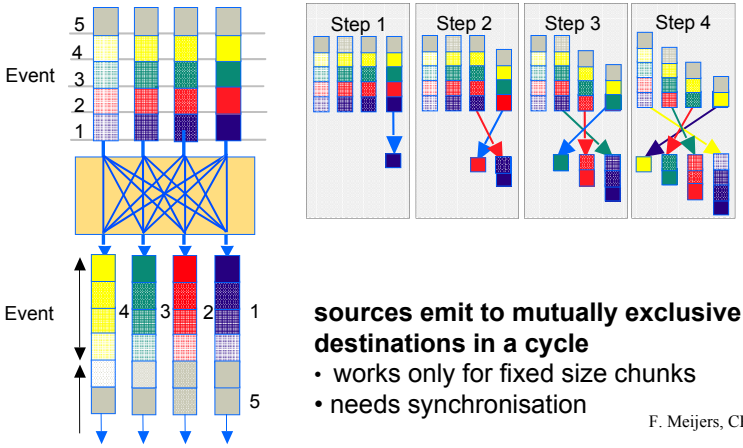


Traffic Shaping (sorting)
usando un ***Barrel Shifter***

L'operazione è sincrona su tutte le sorgenti

Tutti i frammenti di evento devono essere uguali

EVB traffic shaping: barrel shifter

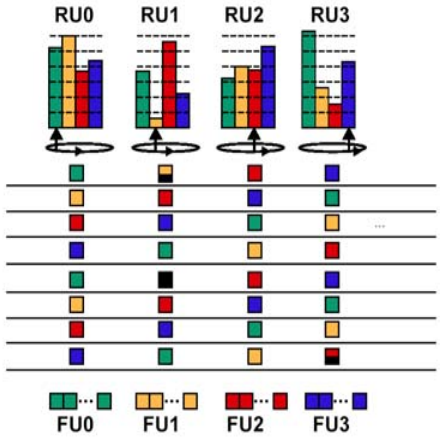


sources emit to mutually exclusive destinations in a cycle

- works only for fixed size chunks
- needs synchronisation

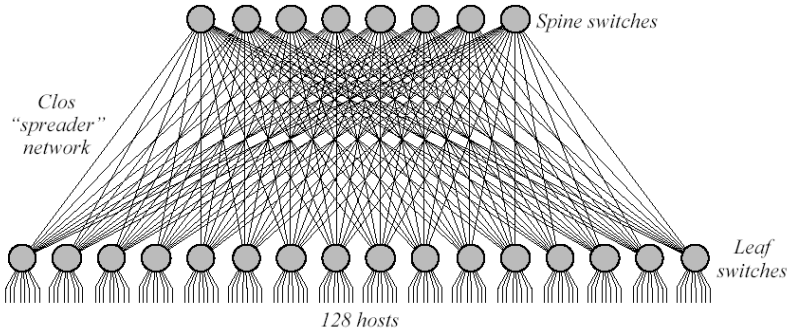
F. Meijers, CERN

Traffic Shaping usando un *Barrel Shifter*



Frammenti di evento di dimensioni diverse sono divisi in pacchetti di dimensione costante. Ci sono pure pacchetti vuoti o riempiti solo in parte.

Uno *switch* di Clos come Myrinet™ è preferibile a un *crossbar*



- richiede *traffic shaping*
- permette connessioni *non-blocking* tra ingressi e uscite (ma.....)
- è *self-routing, wormhole*
- consuma e costa meno di un *crossbar*

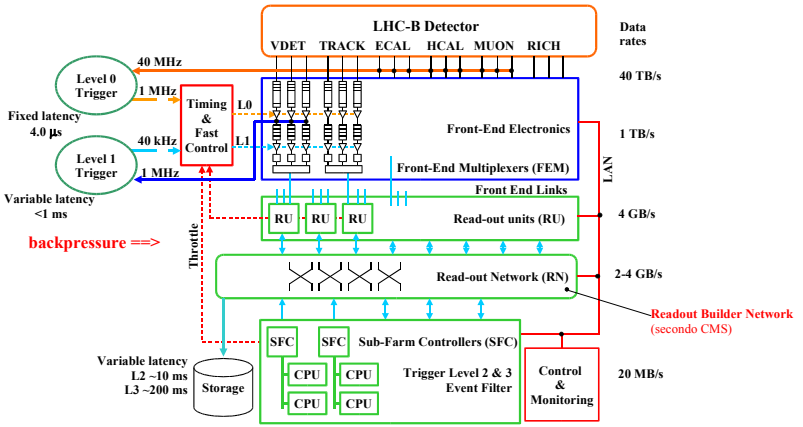
Protocollo *push*

I dati sono spinti alla destinazione da chi li ha generati (sorgente)
La sorgente deve conoscere l'indirizzo della destinazione
Si presume che ci sia abbastanza spazio nei *buffers* della destinazione
Non si può ritrasmettere un frammento di evento
Il protocollo è semplice

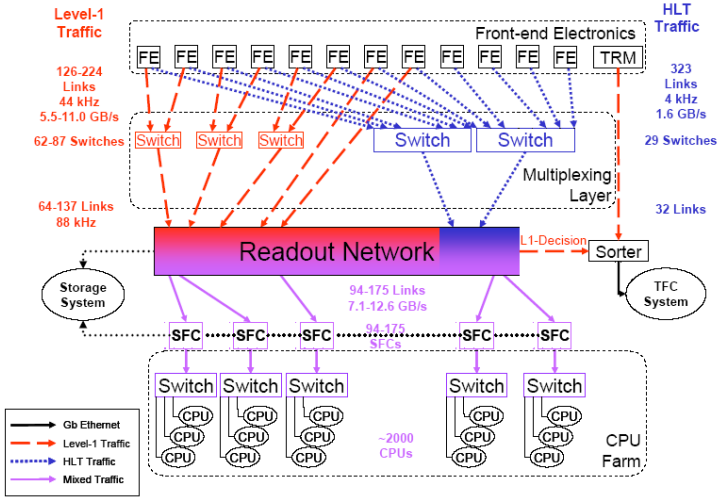
Protocollo *pull*

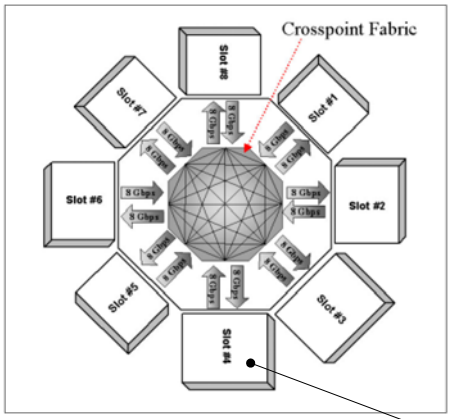
La sorgente deve segnalare alle destinazioni che i dati sono disponibili (*interrupt, look-at-me*)
I dati disponibili sono chiamati, attirati dalla destinazione
Un vero protocollo *pull* può esistere solo in una configurazione basata su *bus*
Le destinazioni possono rileggere frammenti di evento
Le destinazioni devono indicare alle sorgenti quando il trasferimento di un evento è finito, per liberare i *buffers* delle sorgenti
Il protocollo è più pesante

Protocollo *push*: LHCb



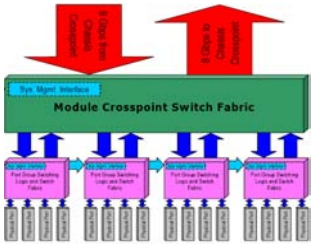
LHCb



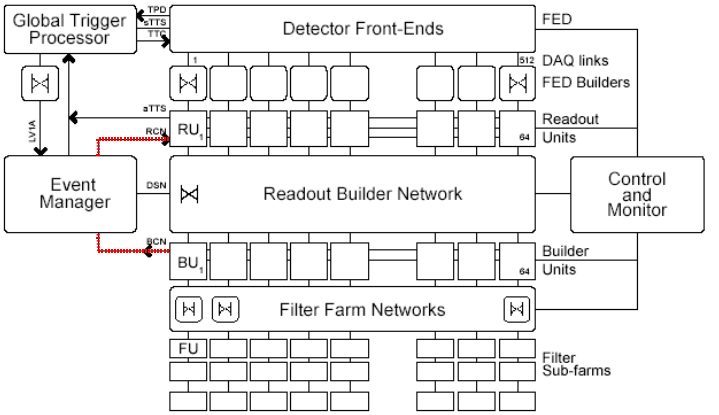


Jetcore, Foundry Networks

Gigabit Ethernet Switch



Prootocollo *pull*: CMS (1)



RCN: Readout Control Network
BCN: Builder Control Network
DSN: DAQ Service Network

CM
S

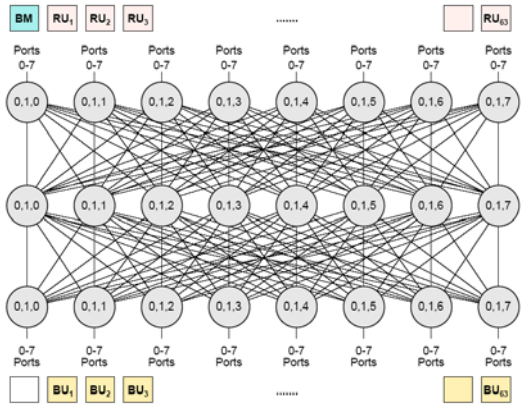
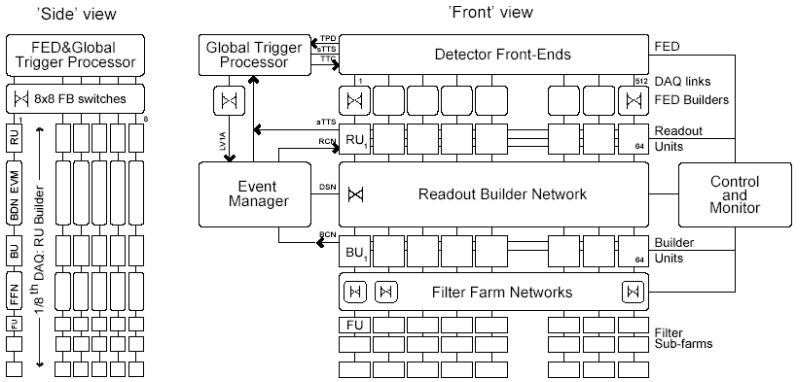
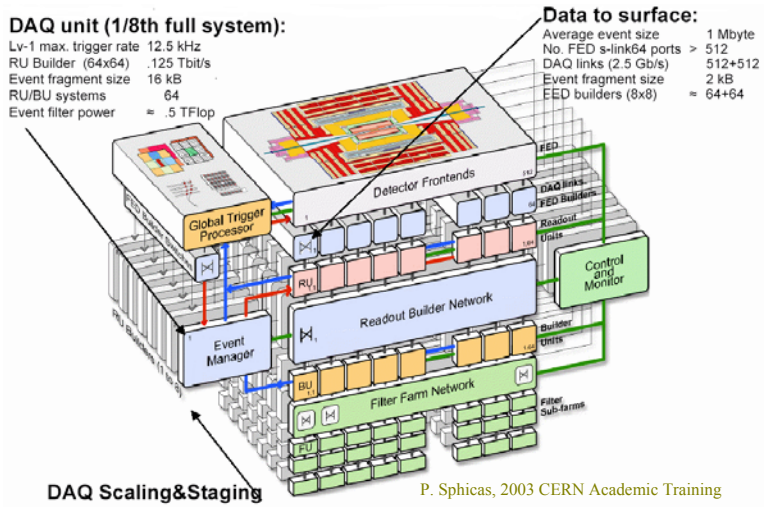


Figure 6-21 Myrinet Clos-128 network used for 63×63 EVB + BM.

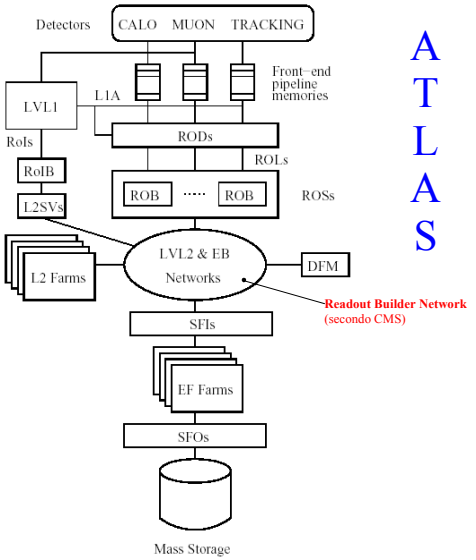
CMS 3D *Event Building (1)*



CMS 3D *Event Building (2)*



P. Sphicas, 2003 CERN Academic Training



ATLAS

Acronyms

DFM	Data Flow Manager
EB	Event Builder
EBN	EB Network
EF	Event Filter
EFN	EF Network
EFP	EF Processor
HLT	High-Level Trigger
L1A	LVL1 Accept
L2N	LVL2 Network
L2P	LVL2 Processor
L2SV	Level-2 Supervisor
LVL1	Level-1 trigger system
LVL2	Level-2 trigger system
OSF	Online Software Farm
OSN	Online Software Network
ROB	Read-Out Buffer
ROBIN	Read-Out Buffer Input
ROD	Read-Out Driver
RoI	Region of Interest
RoIB	Region of Interest Builder
ROL	Read-Out Link
ROS	Read-Out Sub-system
SFI	Sub-Farm Input
SFO	Sub-Farm Output

A
T
L
A
S

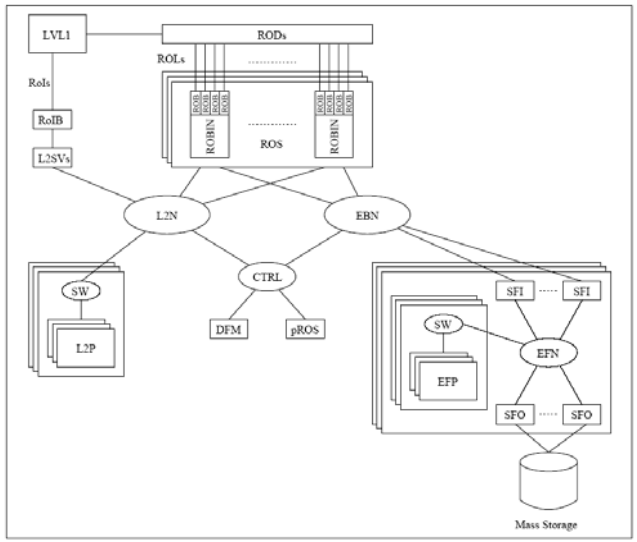


Figure 5-4 Baseline implementation of the DataFlow and High-Level Trigger

Referenze

Per una descrizione delle tecniche di event building:
CMS Data Acquisition and High-level Trigger, Technical Design Report Vol. II, CERN/LHCC 2002-26, CMS TDR 6.2, http://cmsdoc.cern.ch/cms/TDR/DAQ/daq.html

Per simulazione di sistemi di *event building* e acquisizione dati:
The Ptolemy Project, http://ptolemy.berkeley.edu/