

CHEP'07
VICTORIA, BC



**International Conference on Computing
in High Energy and Nuclear Physics**
2-7 Sept 2007 Victoria BC Canada



**LHCb
THCP
ONLINE**

The LHCb High Level Trigger Software Framework

S.Chelukwada, M.Frank, C.Gaspar,
E.van Herwijnen, B.Jost, N.Neufeld, P.Somogyi, R.Stoica

CERN, Geneva, Switzerland

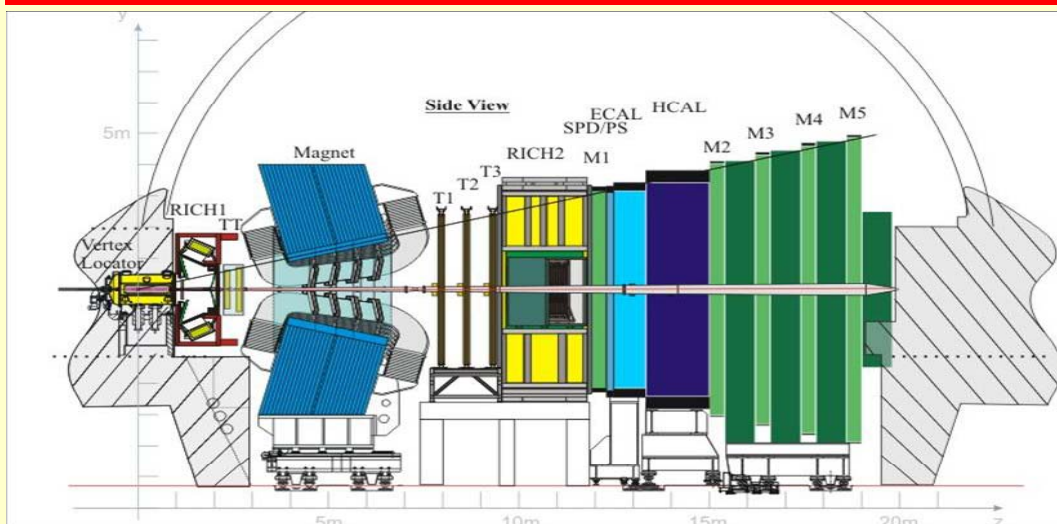


Outline

- LHCb High Level Trigger in numbers
- Motivation which led to our solution
- Basic data processing concept
- Data processing task architecture in HLT
 - Flow of event data
- Implementation considerations



LHCb High Level Trigger in Numbers



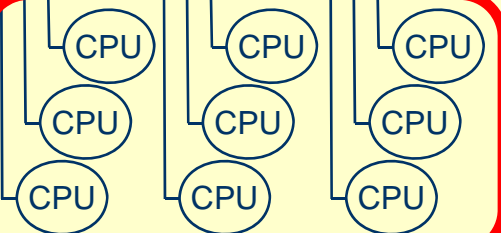
Readout Network

Data Logging Facility

Storage System

Switch Switch Switch

HLT CPU Farm



Region of interest

- Spectrometer for b quark analysis at LHC
- 40 MHz collision rate
- L0 trigger (hardware)
 - Accept rate: ~ 1 MHz
 - Readout NW: ~ 35 GB/s
- HLT (software)
 - Accept rate: ~ 2-5 kHz
 - Event size: ~ 30 kB
 - Data sources: ~ 300
 - Event packing: ~ 10
- ~1000 CPU Boxes envisaged
 - 50 Racks
 - 2000 1U boxes space limit
 - 50 x 12 kW cooling/power limit
 - ~16000 CPU cores
 - ~16000 Trigger processes
 - ~ 4000 Infrastructure tasks



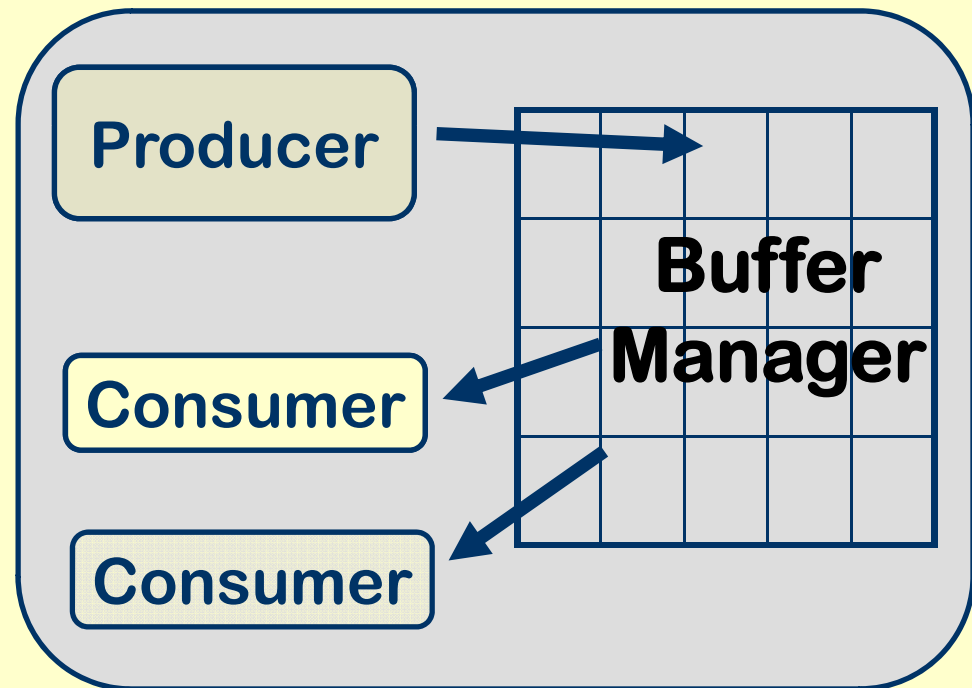
Motivation

- Online – Offline: No longer should the worlds be separated
 - Today's High-Level trigger (HLT) applications are developed in an offline environment.
 - HLT applications are based on the Gaudi framework
 - Boundaries to physics analysis code tend to vanish
 - Applications are then deployed in the online environment
- This requires transparent mechanisms
 - to access event data
 - to collect and transfer event data
=> Poster No. 138 (CHEP 2007, Online track)
 - to collect logger messages
 - to collect performance information: histograms & Co
=> Poster No. 140 (CHEP 2007, Online track)
 - to control interacting processes: startup, initialization, finalization
=> Niko's talk No. 137 (CHEP 2007, Online track)
- ...this will be described in the following slides



Basic Data Processing Concept

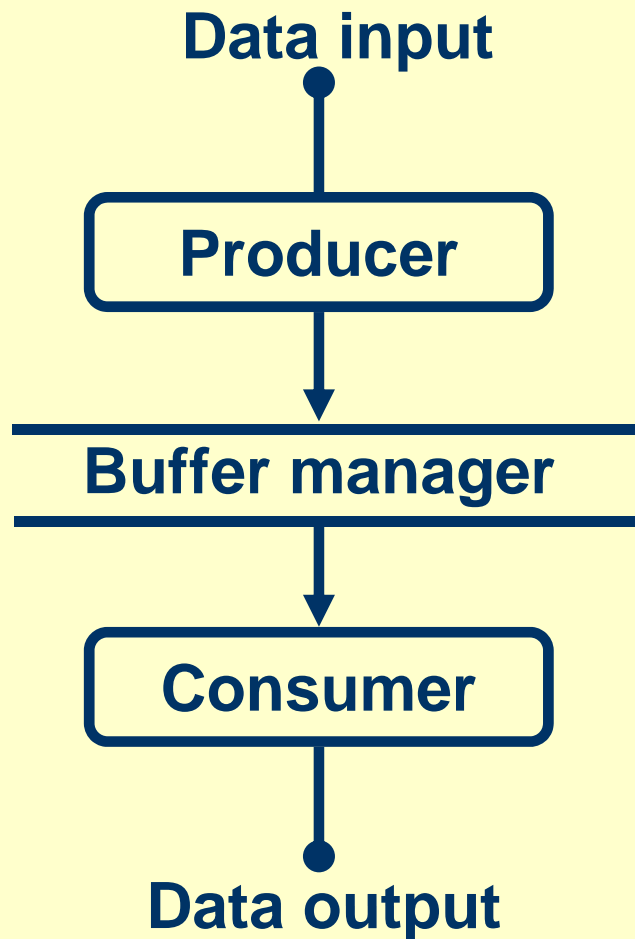
- **Producer Tasks**
 - Receive data from external sources
 - Network
 - other buffer manager
 - Declare data to a buffer manager
 - Optionally process/reformat data [HLT, event assembly]
- **Consumer Tasks**
 - Receive data from a buffer manager
 - Send data to data sinks
 - Network
 - other buffer managers
- **Buffer managers**
 - Data sinks and sources
 - Derandomizing functionality
 - Data selection functionality
 - Multiple buffer managers on each node



A.Belk et al.; DAQ Software Architecture for ALEPH, A Large HEP Experiment
IEEE Trans. on Nucl.Science Vol 36 (Oct 1989); p.1534-1539



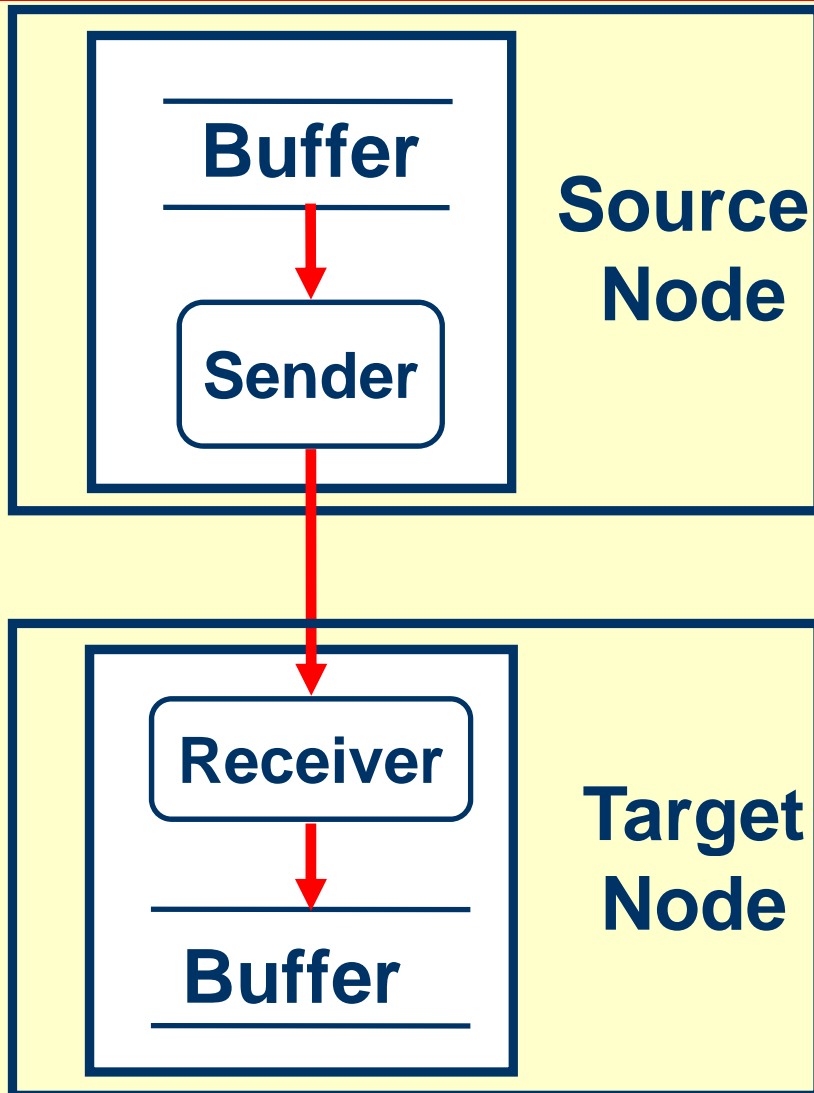
Data Processing Block



- Producers deposit events in buffer manager
 - Partition ID
 - Event type
 - Trigger mask
- Consumers receive events by
 - Partition ID
 - Event type
 - Trigger mask (OR accepted) and VETO mask
 - May queue different requests simultaneously
- 3 Consumer classes
 - **BM_ALL:** Request to receive all events according to request definition.
 - **BM_ONE:** Out of a group of consumers with identical request definition one event is received by exactly one consumer.
 - **BM_NOTALL:** Request to receive some of the events according to request definition and buffer occupancy.



Data Transfer Block

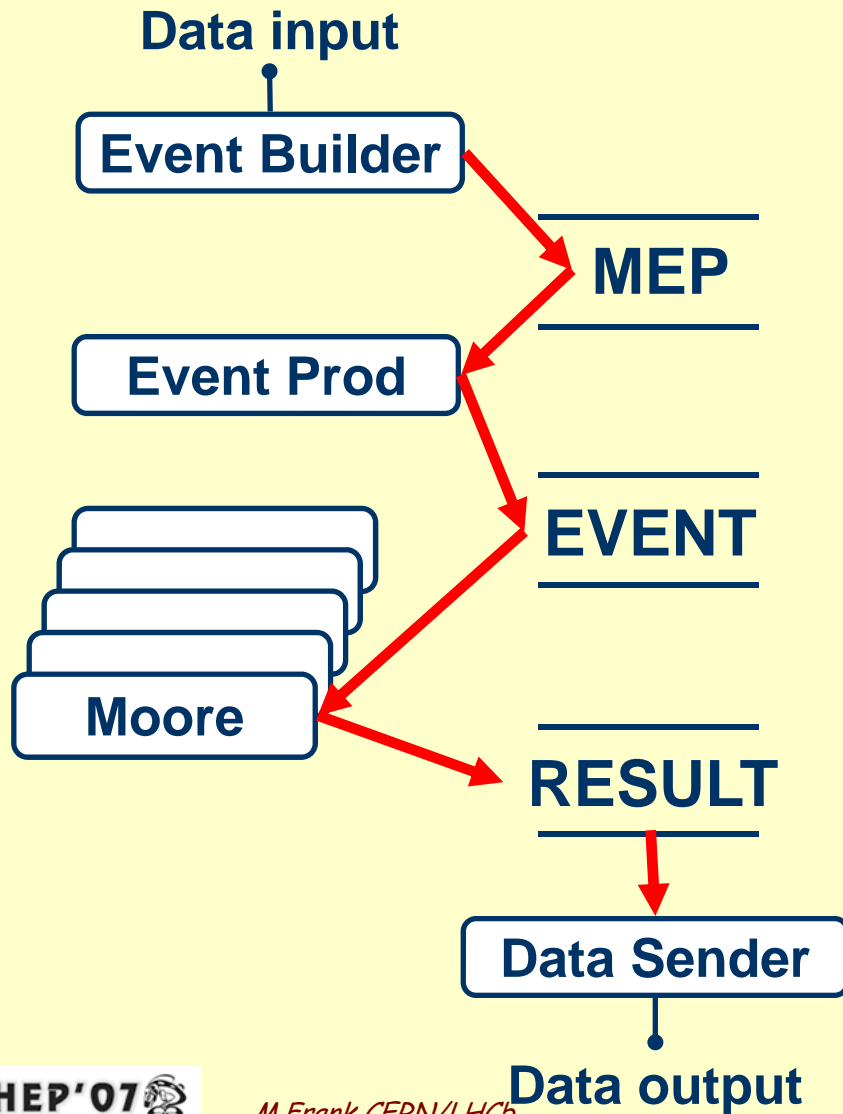


- Reversed data processing block
- Sender tasks accesses events from buffer manager on the source node
 - Consumer process
 - Send data to target process
 - Example: Data Sender on HLT farm node
- Receiver task reads data sent and declares data to buffer manager on the target node
 - Producer process
 - Example: Receiving process on the Storage System

See poster presentation No. 138:
“Data Stream handling in the LHCb experiment”



Task Architecture on HLT Node



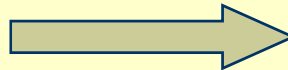
- The **Event Builder** receives the data from the frontend boards and declares a contiguous block to the MEP buffer (N events)
- The **Event Producer** computes N event descriptors and declares them as separate events to the EVENT buffer
- **Moore** trigger processes compute trigger decision and declare accepted events to the RESULT buffer
- **Data Sender** tasks send accepted events to the Storage System



MEP Buffer and Event Descriptors

- Store data once, pass references to banks from frontend boards
- Important optimization to avoid many expensive, unaligned memory copies
 - LHCb data / frontend source: ~ 100 Bytes => event size: 30 kB
 - very ineffective for DMA transfers

Multi event buffer block



Descriptors with single events

Multi Event: packing factor m			
Source ID 1	Evt 1	...	Evt m
Source ID 2	Evt 1	...	Evt m
...
Source ID n	Evt 1	...	Evt m

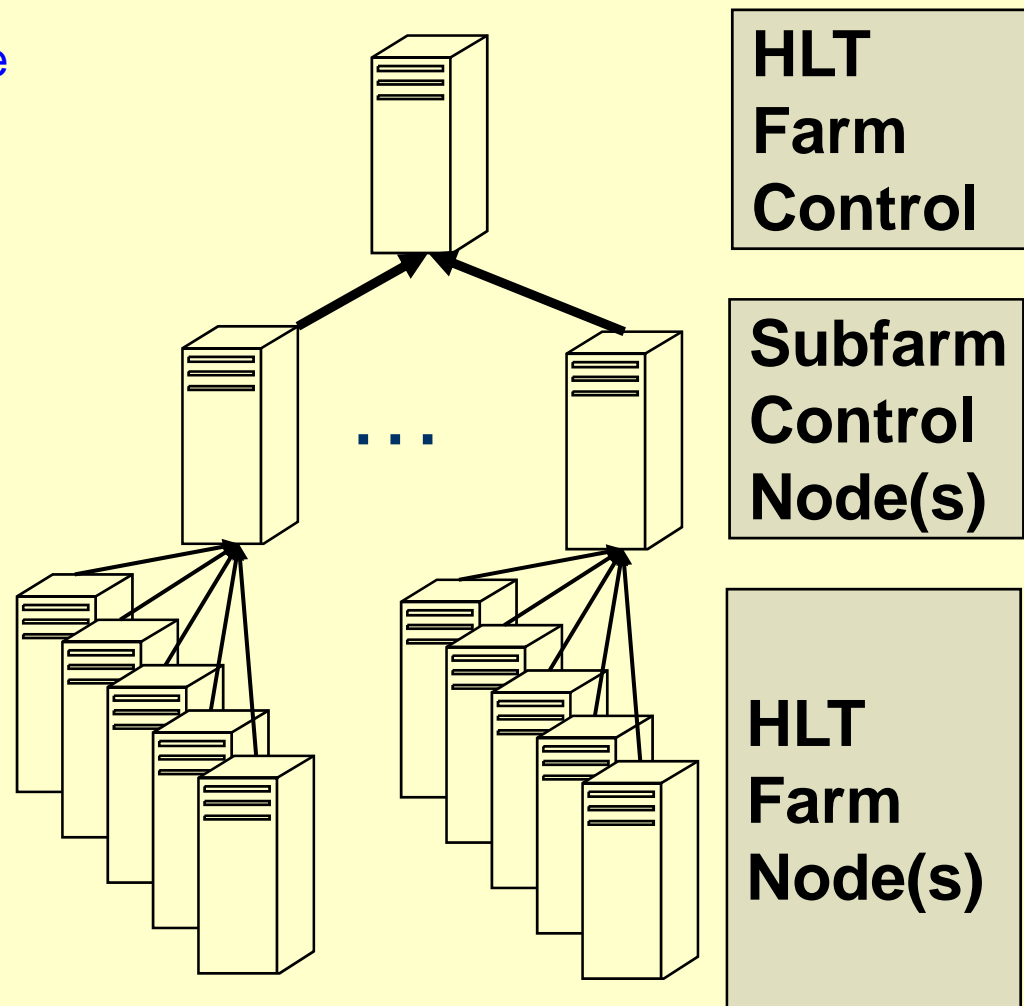
Event # 1		
Ptr	Source ID 1	MEP 1
Ptr	Source ID 2	MEP 1
...
Ptr	Source ID n	MEP 1

Event # m		
Ptr	Source ID 1	MEP 1
Ptr	Source ID 2	MEP 1
...
Ptr	Source ID n	MEP 1



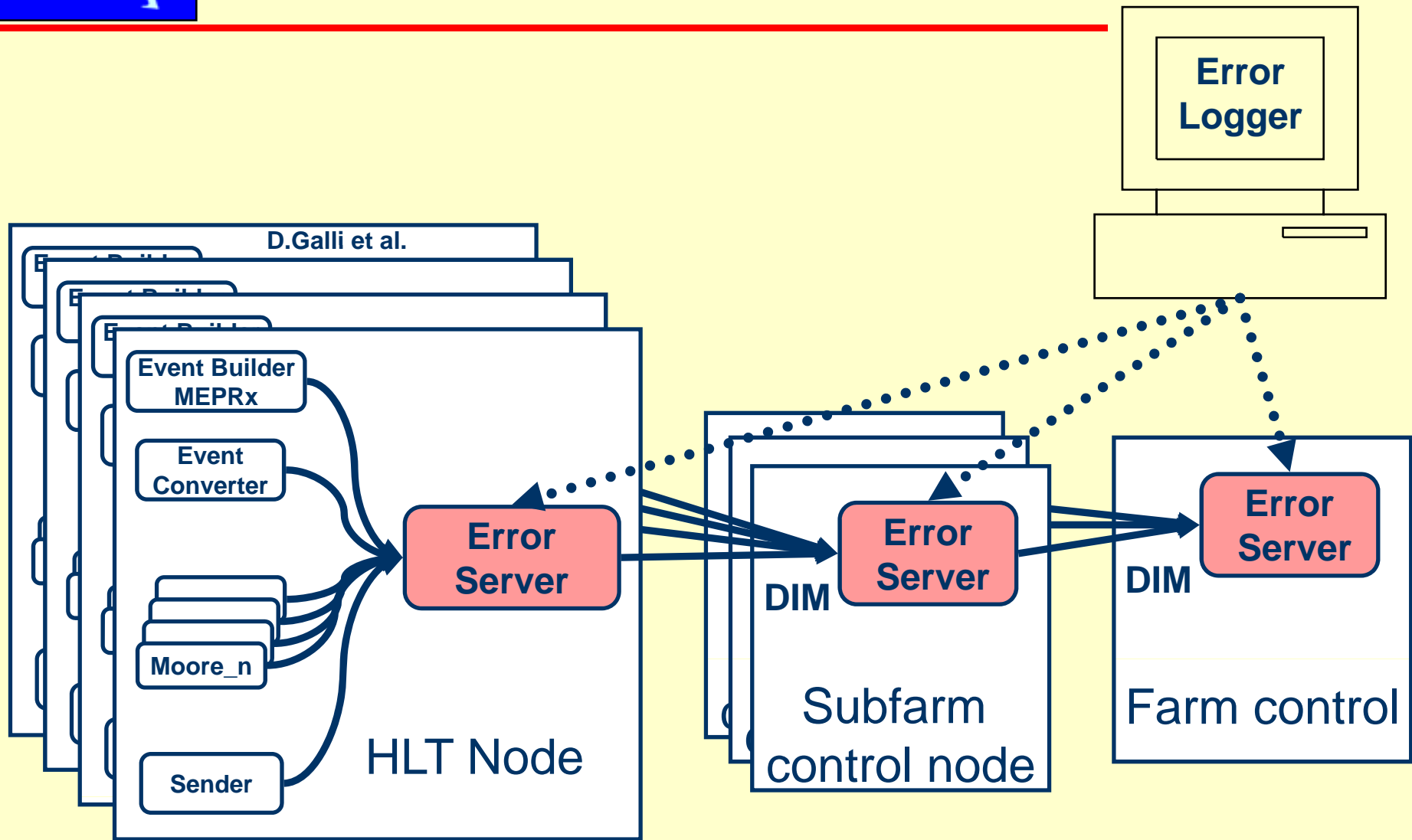
Error & Output Logging

- Every tasks tends to print the bible
 - x 16000 => cannot be managed
 - Need to restrict output
- Hierarchical approach
 - HLT Farm
 - Subfarm - Farm Node
 - Storage network
- Filtering at each layer
 - Accept/refuse messages
 - By task name (UTGID)
 - By component/algorithm name
 - By node
 - By message content
 - Wildcard selection
- Intercept at each level



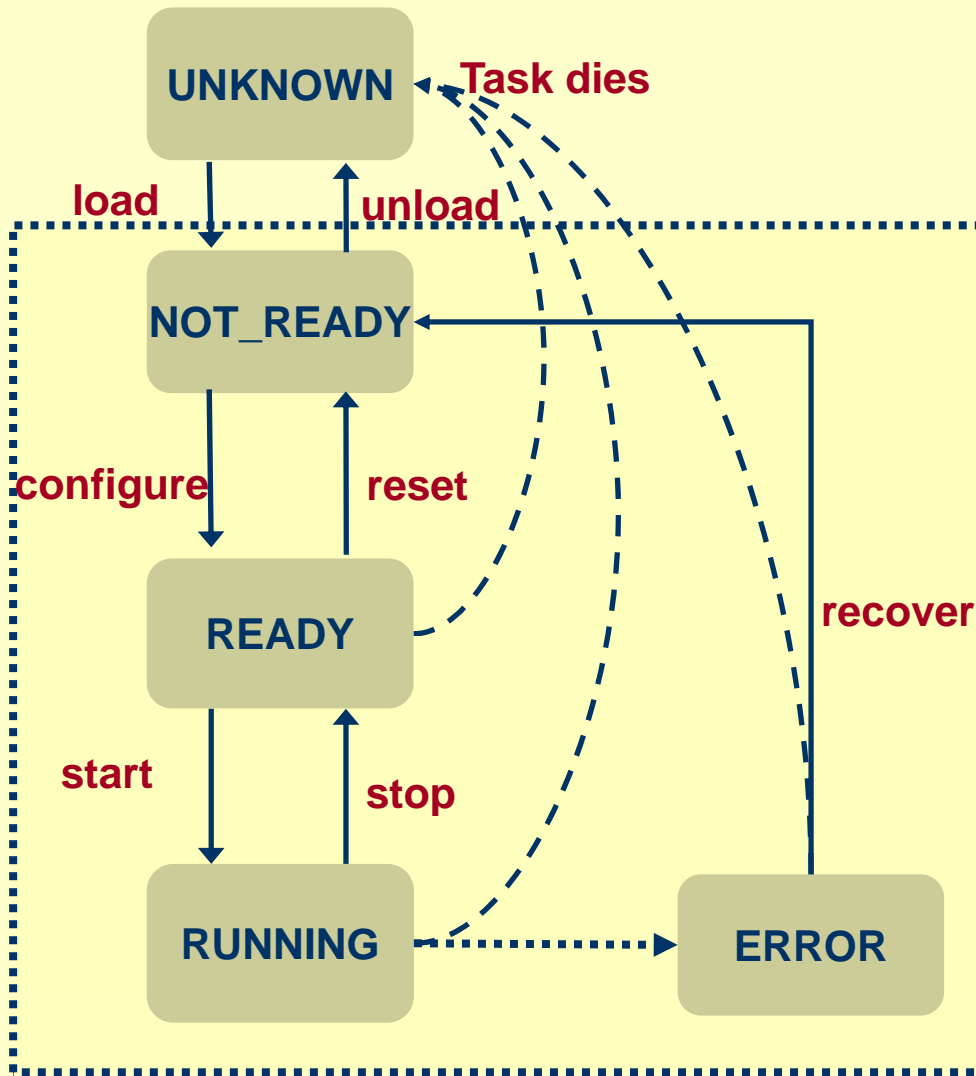


Error & Output Logging (2)





Task Control



- All tasks are based on Gaudi data processing framework
- Common state diagram as shown
- Control using common Experiment Controls System [ECS] based on DIM / PVSS
- Transitions are mapped to Gaudi transitions
- Satisfies required functionality for:
 - Infrastructure tasks: buffer managers
 - Data processing tasks: event builders, HLT filters/Moore and data transfer tasks



Implementation Considerations

- All applications are implemented as Gaudi tasks
 - Ease offline development of HLT algorithms and online deployment
 - Event filter algorithms are identical in offline and online
 - The offline application is executing in the online environment
 - Some services were replaced/adapted to the online:
 - Identical interfaces
 - Event access using buffer manager
 - Message reporting utility
 - Dynamic application bootstrap
- All components available in 2 shared libraries, which are loaded at image activation time
- All OS dependencies are encapsulated
 - Supported platforms are WIN32 (development/debugging) and linux (deployment on HLT farm)



Conclusions

- We developed an open framework to execute HLT code in the LHCb online environment
- No online-offline separation
 - Transparent data access
 - Transparent data transport
- Both realized using very simple building blocks
 - Buffer managers
 - Common control structure interfaced to Experiment Control System which uses PVSS
 - Networking library to transfer data between processors
- All OS dependencies are encapsulated