DAQ@LHC
HLT Framework Plans

Outlook after LS1

Input from:
ALICE: T. Breitner
ATLAS: W.Wiedenmann
CMS: E.Meschi
LHCb: M.Frank, C.Gaspar, B.Jost

Outline

- Short discussion about the various designs
  - Emphasis in process architecture in the filter farm
- Solutions to common problems
  - Output logging
  - Histogram collection / presentation
- Special topics
  - Fork & COW
  - Checkpointing
  - Deferred event filtering
- Emphasis on expected status after LS1

To Give You an Idea: System Scale

Number of..	Boxes	CPU cores	Filter procs	Logical Grouping
ALICE	~ 200	~ 5000 (1)	~ 3000	
ATLAS	~ 1600	~ 17000	1 per core (2)	49 Racks
CMS	~ 1600	~ 16000	~ 35000 (2)	O(20)BUs in 8 slices
LHCb	~ 1600	~ 16000	~ 30000 (2)	57 Racks

(1) 2300 CPU cores + 54 FPGA + 64 GPU cards (estimated to 100-200% of the CPU)
(2) Overcommitment if hyper-threading is supported by worker node

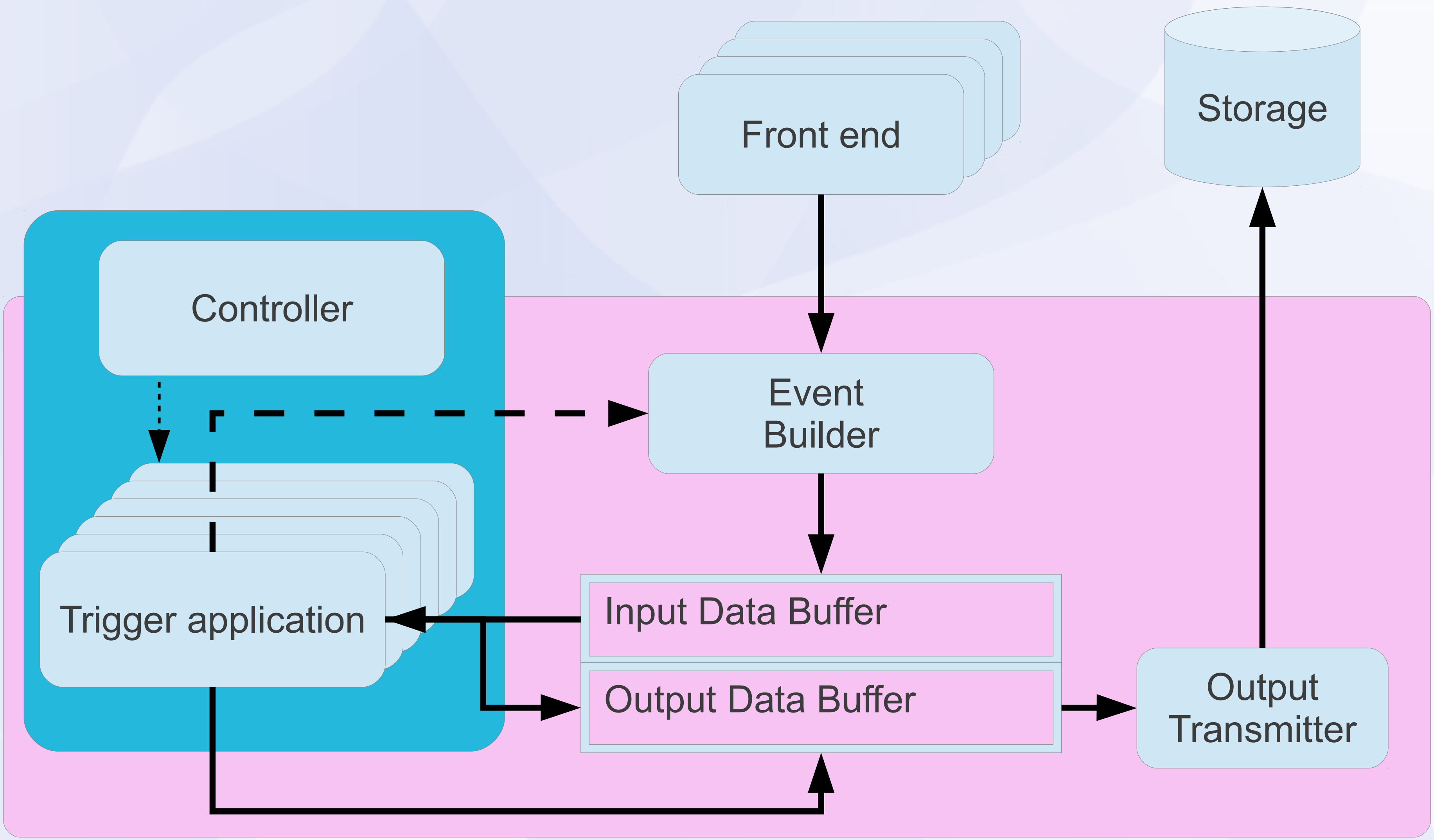
Architecture

- Logical Architecture
  - Nodes and processes
- Hardware Architecture
  - Basics only
- Software related issues
  - Event data transfer within worker node
    Shared memory
  - Data transmission protocol
  - Data exchange format

Architecture:
The diversity between experiments

- To access and transport event data all experiments are at the end limited by the constraints of the operating system (Linux only)
  - shared memory
    [used to share event data between processes]
  - network connections
    [used for data transfers]
- The various different approaches show high level of creativity
  - Leading to quite different solutions

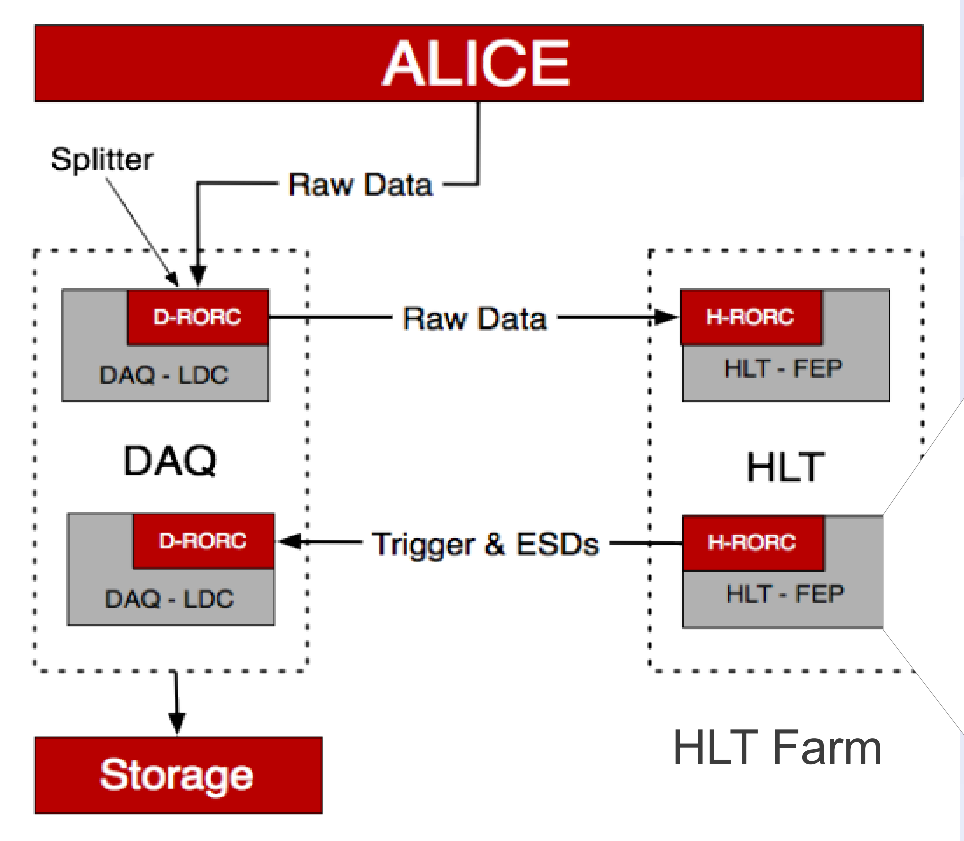
Logically: The Basic Problem



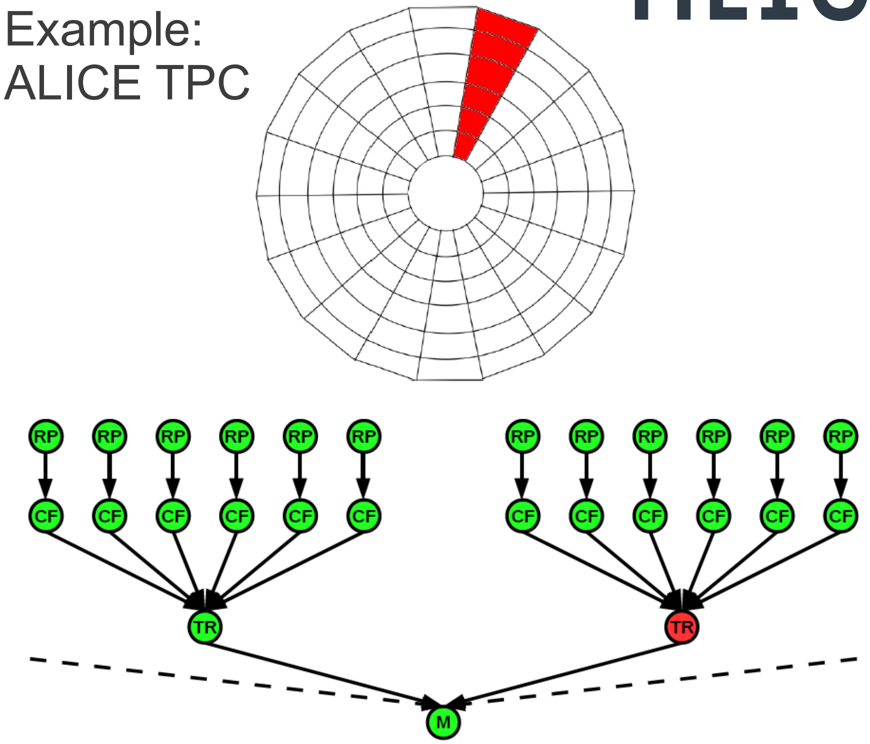
So far the theory

- This logical decomposition
- … obviously leads to various different implementations concerning
  - Process control
  - Event data access
  - Propagation of the HLT output
- Let's have a closer look
  - On what is planned after LS1

Data Processing with “Algorithm Pipelines”



Example:
ALICE TPC



- Chains of processing elements
- Iterative data merging
- Process based parallelism

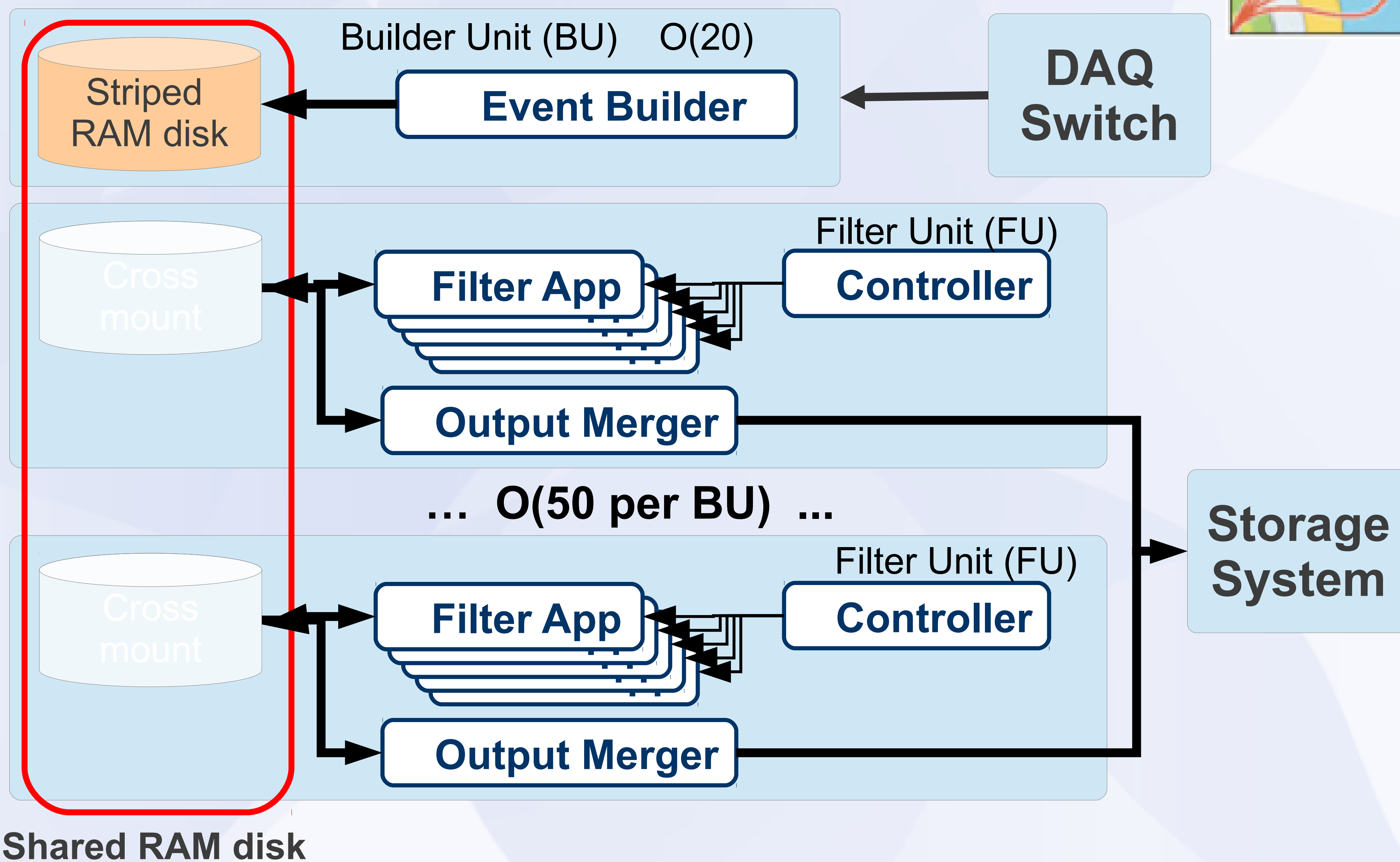
Algorithm Pipelines
An entirely different approach

- Hardware wise
  - Data are duplicated by DAQ
  - HLT data are sent back to DAQ (like subdetector)
- Software wise
  - A parallel approach to a parallel problem
  - The problem all other experiments try to solve now using a multi-threaded approach
  - Are here done using specialized processes
- Ideas remind me of 'Iris Explorer'

Algorithm Pipelines Technicalities

- Today: 1 event is handled on 1 node
  - Implementation allows to handle single event on multiple nodes, but not necessary
- Event builder receives fragments from DAQ
  - And merges the arriving fragments in steps
- Processes handling event data
  - Communicate using fifos
  - Pass data via shared memory
- Data writer sends HLT results back to DAQ
  - No event filtering: online reconstruction and compression to reduce data volume

“Offline Oriented”



Features

- **Approach triggered by hardware move from Myrinet to Infiniband based NICs**
  - **High bandwidth, but limited switch ports available => O(20) builder units**
- **Aim: Complete software decoupling of online and offline components**
  - **Ease of sw release cycles**
    - **Trigger application built by offliners**
    - **Merger and event builder purely online**
  - **Nice: If something does not work, there is a clear victim to yell at...**

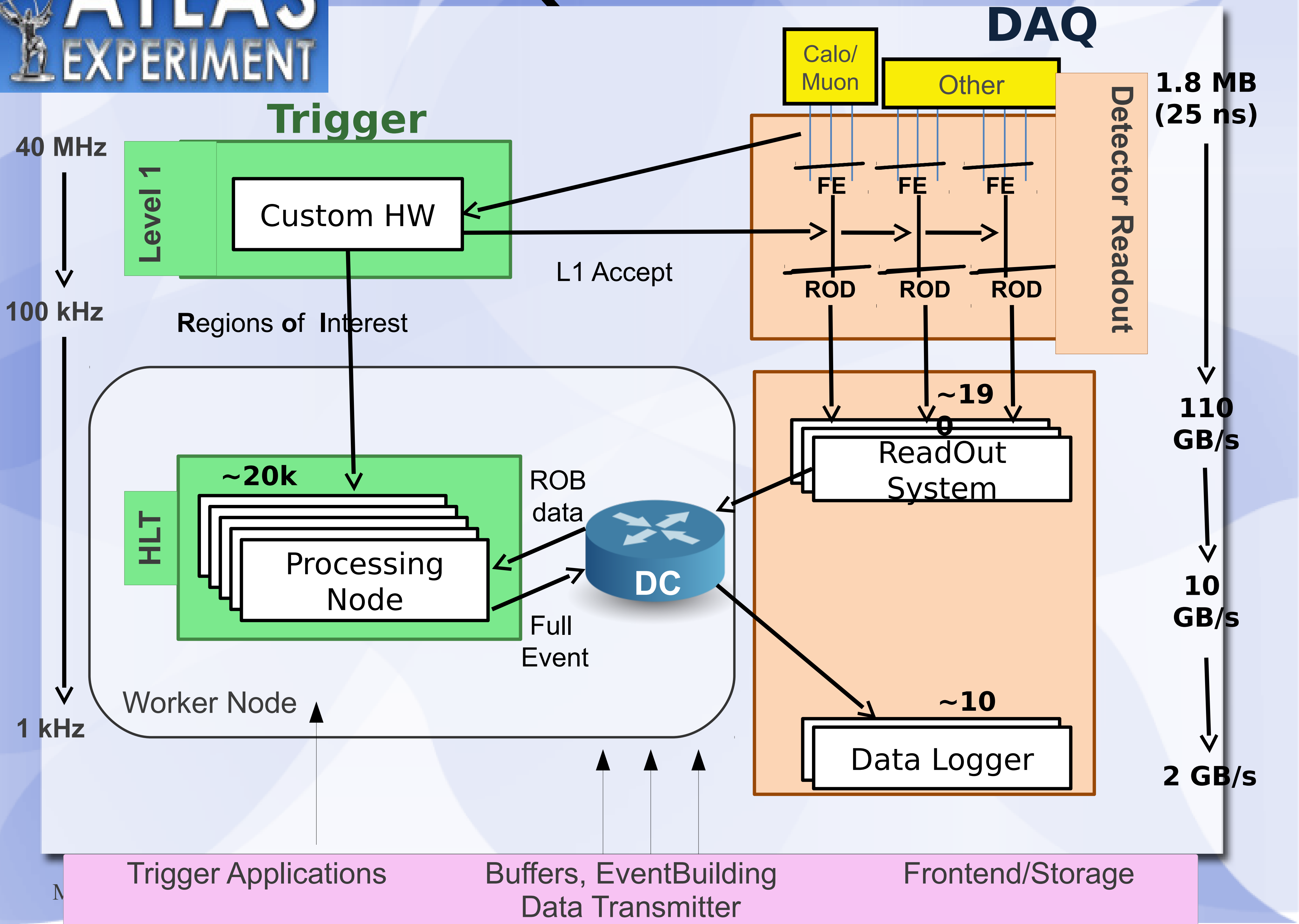
Features

- **Data are accessed through shared file system**
  - **Let the operating system do the job**
  - **Effectively no more messaging in offline applications**
  - **Processes 'poll' on the occurrence of new files**
  - **Output files contain event which belong to one lumi section (2^{18} orbits)**
    - **On worker: O(10) event output files per trigger app**
    - **These files are concatenated on multi-stage logger nodes**

Advantages

- CMS has a bank based data format like LHCb
  - Simple file handling: merge = elaborated 'cat'
- The operating system does much of the job
  - No more events input/output management in the BU
    This is implicitly done by the file system
  - But not for free: shared ram disk has throughput of 2 GB/sec both write (BU) and read (FU)
- Cascaded merging process of accepted events
  - On the filter unit
  - Multiple output streams

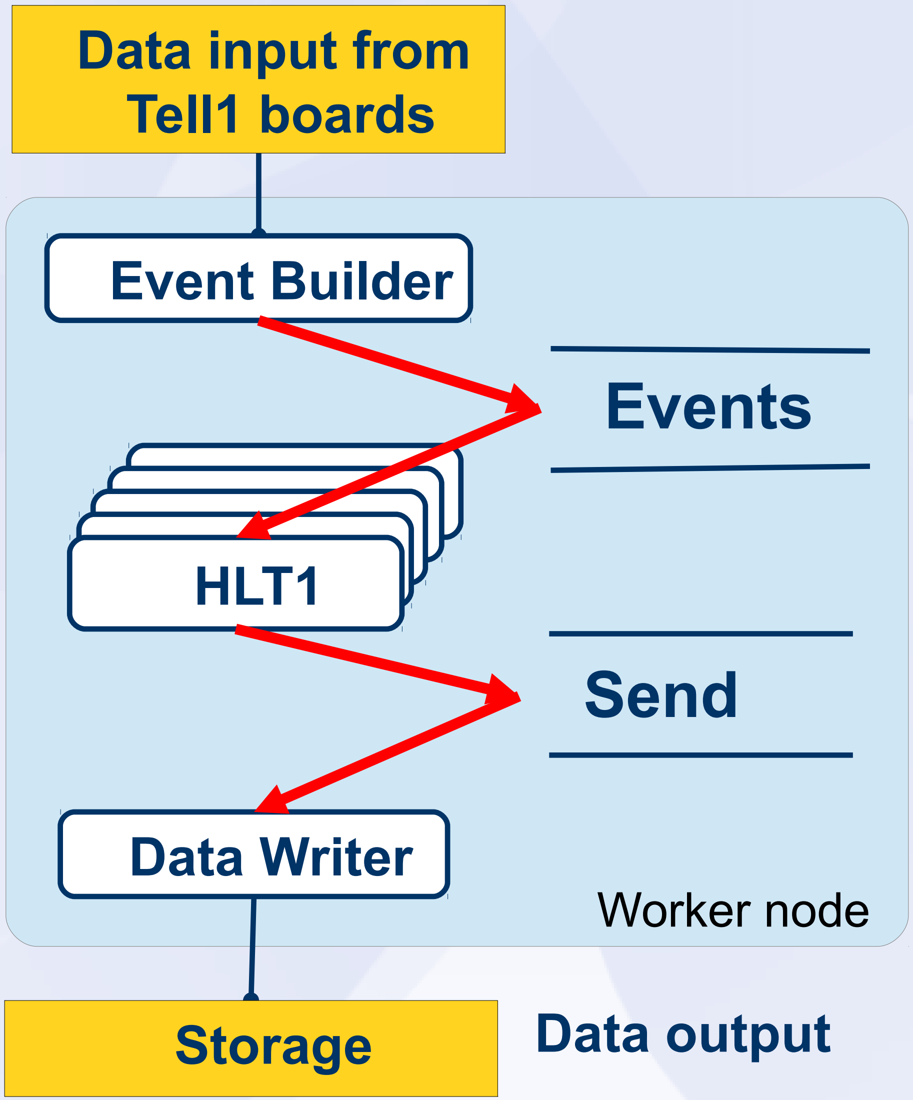
TDAQ After LS1



Highlights

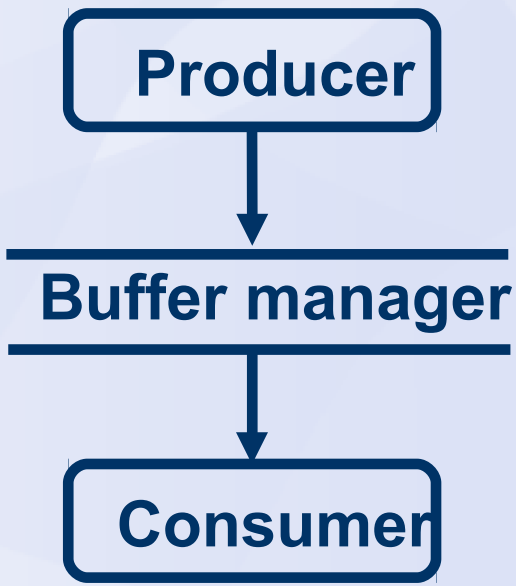
- Significant simplification compared to Pre-LS1
- Processes are controlled by RC at the node level
- Data Collection Manager (DCM) is personal union
  - Buffer manager, “Event builder”, Data Transmitter
- Trigger applications forked from single parent
  - Memory benefits from copy-on-write
- Trigger applications request data by piece from DCM
  - Specialty: Event data are pulled
- Event data reside in shared memory
  - Input data on request from DCM
  - Output data managed together with input

"Cascaded Buffers"



- **Event Builder**
  - **receives the data from the front-end boards**
  - **declares a contiguous block to the *Events* buffer (N events)**
- **HLT1 trigger processes**
  - **compute trigger decision**
  - **declare accepted events to the *Send* buffer**
- **Data Writers send accepted events to 'Storage'**

Single Processing Pattern



- Producers deposit events in buffer manager
- Consumers receive events
- Pattern reoccurring everywhere
- e.g. Trigger applications are both
  - Consumers and Producers
- Forking applied (COW)
  - Memory reduction > 80%
- Processes steered by on each node by a 'node-controller' managed by PVSS/SMI

Unfortunately: This is not the whole story

- After LS1 'deferred triggering'
  - Same patterns, slightly different usage
  - I will come back to this later

Outline

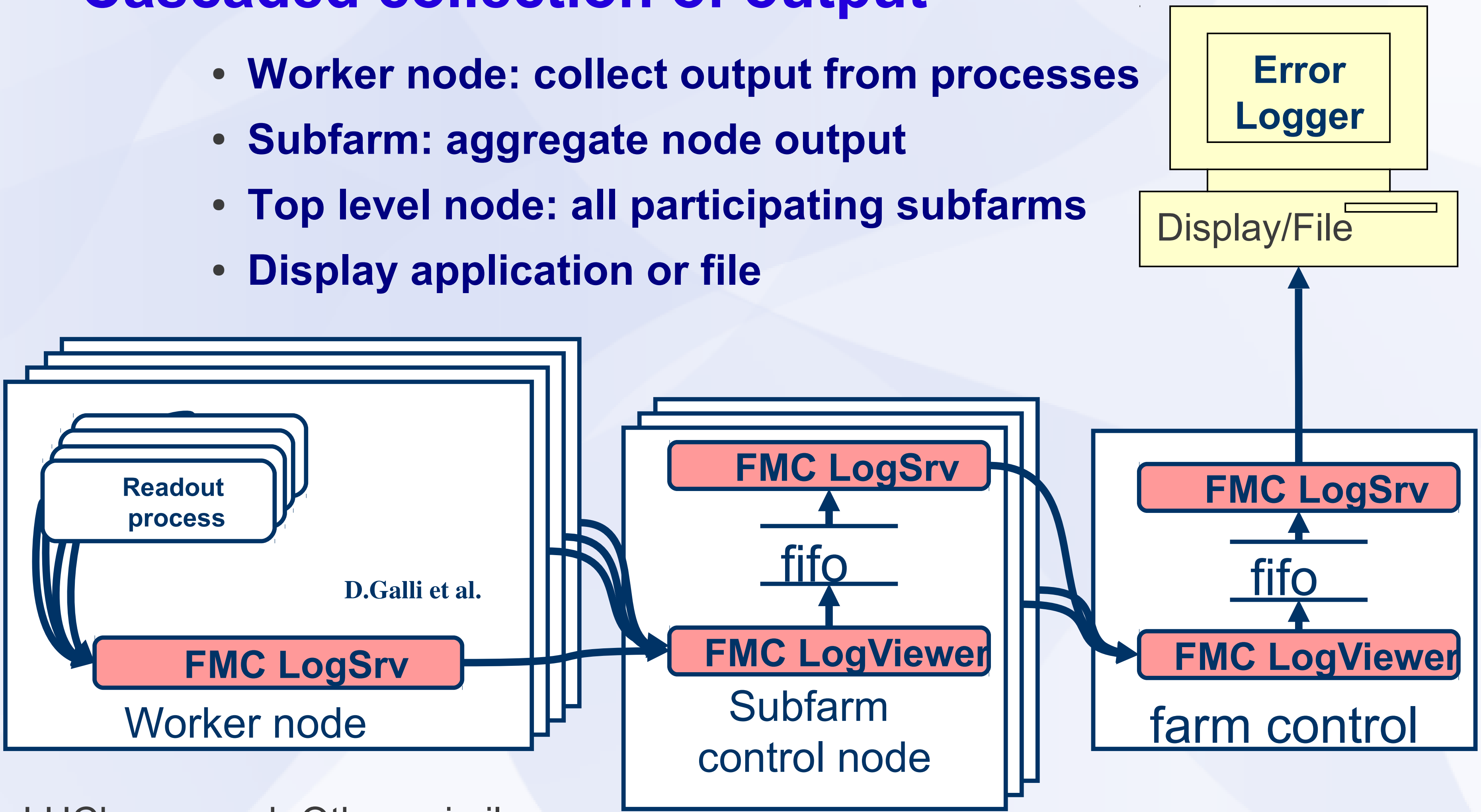
- Short discussion about the various designs
  - Emphasis in process architecture in the filter farm
- Solutions to common problems
  - Output logging
  - Histogram collection
- Special topics
  - Checkpointing
  - Deferred event filtering

Implementation

- **Issues loosely connected to event processing**
  - **Printout/Message collection**
  - **Histogram collection**
  - **Crash handling**

Error & Output Logging

- **Cascaded collection of output**
  - **Worker node: collect output from processes**
  - **Subfarm: aggregate node output**
  - **Top level node: all participating subfarms**
  - **Display application or file**



LHCb approach Others similar

Implementation: Error & Output Logging

- **Common problem: Cascaded collection of output**
  - Led to separate implementations throughout the experiments. Example transmission protocols
  - LHCb: DIM
  - CMS: log4j based proprietary protocol
  - ATLAS: proprietary protocol
  - ALICE: n/a Output collection only internal to HLT Presented to the shift crew are only summaries and state information
  - Typically the messages are kept for several weeks

Implementation: Error & Output Logging

- **Common problems, individual solutions**
  - **Logging to graphical output device for shift crew**
  - **Logging to file**

- **Problems due to large # of identical processes**
  - **Suppression of duplicated/similar messages**
  - **Is such trouble addressed at all ? [LHCb did not...]**

Histograms & Counters

- Similar problem like collecting output
- ALICE:
  - n/a pipelined algorithm approach
    Some selected histos and counters presented to the shift crew for comparison with reference
- ATLAS: Histograms are collected from all HLT apps
    - Separate readout tree.
    - Merged in a dedicated histogram gatherer
    - Subset is presented to the shift crew for comparison with corresponding references.
  - Selected counters and rates are presented in form of time charts

Histograms & Counters

- CMS: Collected in each process, then written and shipped with the same mechanism as event data
  - Some counters stored to database
- LHCb: Similar to ATLAS
  - Separate readout tree using DIM publishing
  - Some counters go to PVSS archive
  - Some rates are trended

Crash Handling

- ATLAS/CMS store events, that caused a crash in their raw format for subsequent analysis
- CMS in addition keep core files
- LHCb: Possibility to enable the collection of core files

Outline

- Short discussion about the various designs
  - Emphasis in process architecture in the filter farm
- Solutions to common problems
  - Output logging
  - Histogram collection / presentation
- Special topics
  - Fork & COW
  - Check-pointing
  - Deferred event filtering

Fork & COW

- 3 experiments benefit from copy-on-write (COW)
  - Large parts of memory are written once and only accessed in read-only mode (or never → amount of zero-pages)
  - Magnetic field maps
  - Detector description (geometry, parts of alignment,..)
  - Also other memory section are only initialized once
- Memory is reused by OS between related processes
  - LHCb: up to ~80 % mem saved, CMS ~ 40 %
- Not needed by ALICE: different approach
  - ALICE does not have thousands of identical processes

Fork & COW

- COW implicitly requires forking. 2 approaches:
  - Fork() only clones the main thread
    Keeps file handles shared
  - Either allow 'atfork' handlers (Atlas/CMS)
    - Stop all threads and close all files/connections before fork
    - Restart all threads and reopen all files/connections after fork
    - Beware of Oracle & Co.
  - Do it behind the scene (LHCb)
    - Only works for 'real files'
    - Still a bit tricky: restore temporary (already deleted) files
    - Cannot handle network-, oracle- and other connections
    - Threads are restarted from the pc they were halted

Fork & COW: Plans

- **ATLAS, CMS, LHCb**
  - Because we have so many identical applications copy-on-write works very well
  - 'In principle' convinced to be able to handle also > 32 cores
    LHCb: small events O(100) processes/node possible
    Atlas: O(100) may still work
  - True limit unclear, but some agreement, that it will take time until hit
  - No in-process parallelism planned in near future though off-line is looking into it
- **Beyond 100 cores trouble may start: after LS2 ?**

Process Restart from Checkpoint

- **HLT configuration is lengthy**
  - **Difficult to reduce**
  - **Would require intrusion of offline code**
- **Cold start: CMS O(1min), LHCb/ATLAS O(>5min)**
  - **Reason: Processing detector description / conditions**
  - **CMS is faster, suffer at each run change O(30sec)**
    - **When new conditions enter**
    - **Still further plans to reduce this time**
  - **'Everybody suffers at changing pain levels'**
- **Problem does not state itself for ALICE**
  - **Different processing model** (built-in parallelism)

Process Restart from Checkpoint

- **Why not restart from a 'core-dump' ? (*)**
  - **Load already configured image from disk**
  - **Post-configure step**
  - **Run ...**

(*) J.Ansel, G.Cooperman, M.Rieker,
Transparent User-Level Checkpointing for the Native POSIX Thread Library for Linux,
The 2006 International Conference on Parallel and Distributed Processing
Techniques and Applications (PDPTA'06), Las Vegas, NV. Jun., 2006.

Checkpoints

- Save process image to file
  - Save all open file descriptors
  - Halt all threads at a well defined position, so that the thread can be recreated and the instruction pointer set to this location
  - Save all memory mappings
- Restore process image from file
  - Restore file descriptors
  - Restore all memory mappings (libs+heap)
  - Restore stack
  - Create threads and set saved instruction pointer

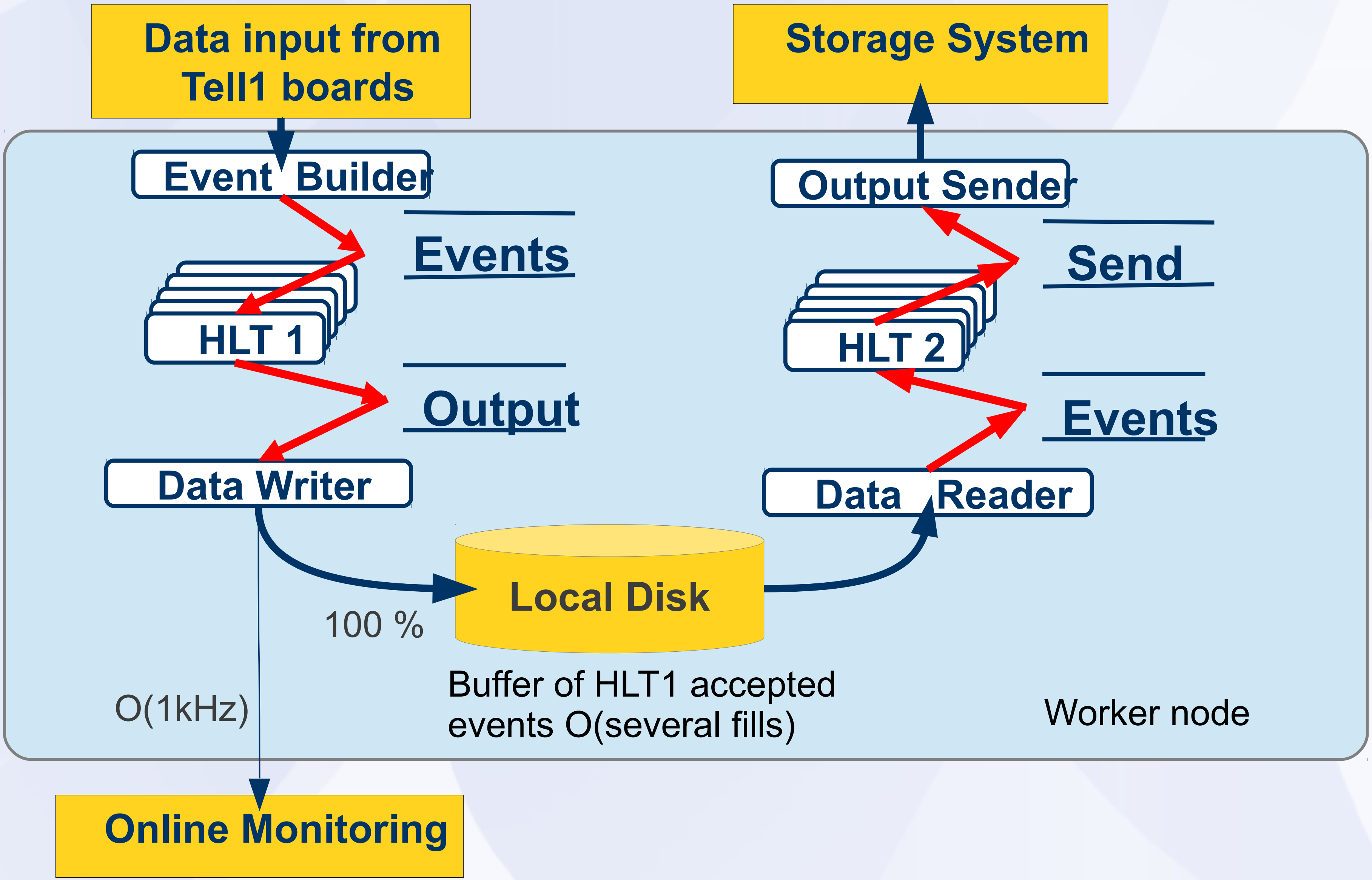
Checkpointing

- Checkpoints are a vehicle to navigate around the problem of long HLT process initializations
- We have made good experiences
- Requires maintenance
  - OS / GLIBC upgrades, etc.
- Checkpoint file distribution was problematic
  - Distribute ~2.5 TByte within 'seconds'
  - Solved using Bit-torrent approach

Intrinsic Sociological problem:

In the presence of checkpointing the motivation to resolve the original problem is much smaller

Deferred Trigger



Deferred Trigger

- Full use of the CPU power of the HLT farm including inter fill gaps and technical stops
  - Better event selection
    accepted events enriched with 'good' events
  - LHC delivers only ~30% of the time 'Stable Beams'
  - Possible boost of CPU power up to factor 3
  - Data of several fills in local disk buffer
    - HLT 1 largely reduces number of events to be saved locally
  - Control of HLT1 and HLT2 is entirely decoupled
    - Like 2 seperate DAQ systems
    - One with and one without detector

Deferred Trigger

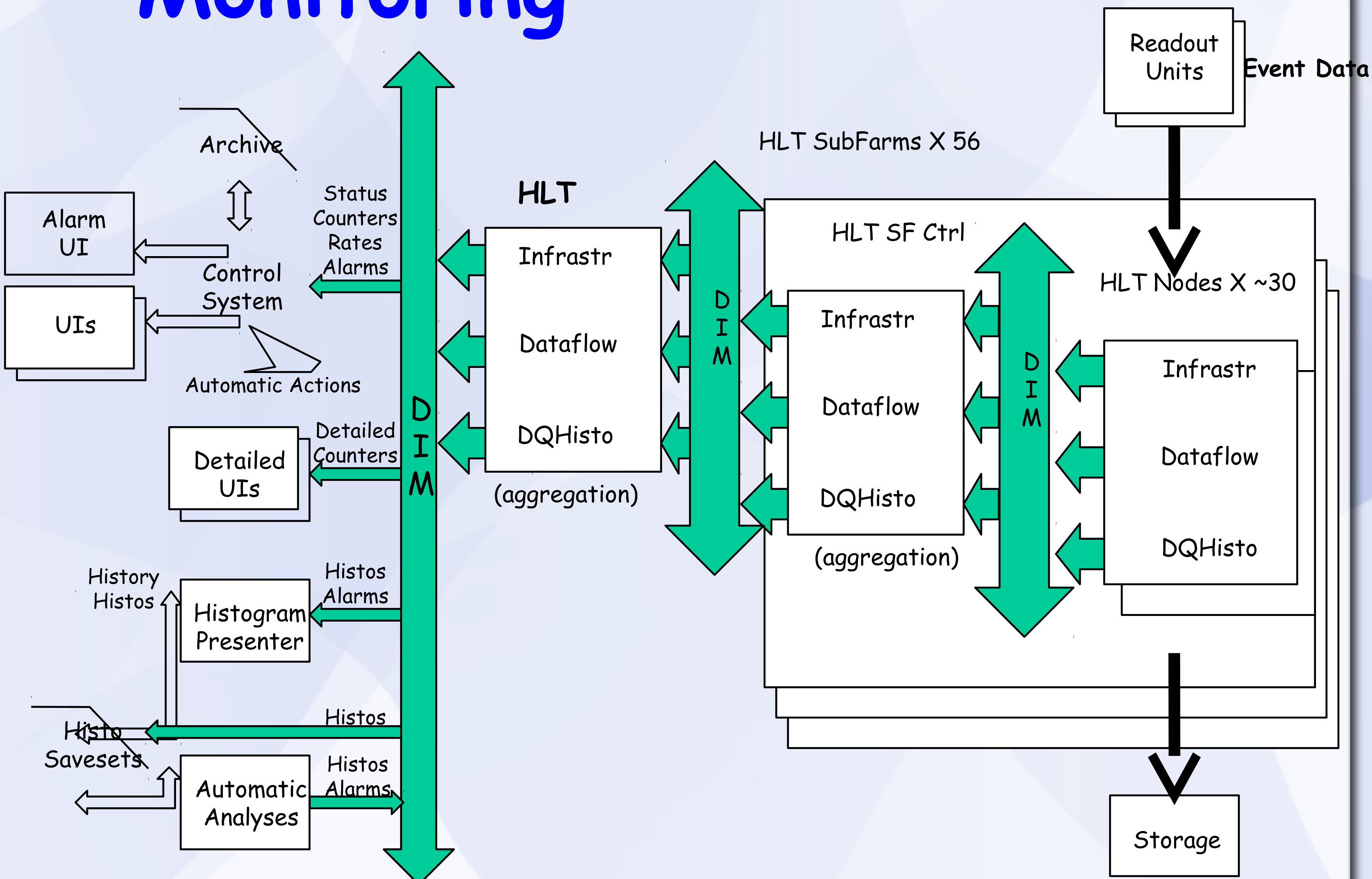
- **ALICE: n/a**
  - **HLT is handled like subdetector**
  - **Must fulfill latency requirements**
  - **Plans to use HLT farm for offline jobs**
- **ATLAS: Possibilities for implementation under study. No decision yet**
- **CMS: Can buffer few minutes. Otherwise at the HLT input too large events and too high rate**
  - **Online HLT farm designed for peak usage**
  - **Offline processing planned to use CPU capacity**

Conclusions

- **Many similar problems were solved individually**
  - **Because detectors *are* different**
    - **Event data pull (ATLAS) vs. data push (CMS/LHCb) or**
  - **Different HLT architecture (ALICE)**
- **This divergence continues today**
  - **Seen various different plans for the future**
- **It looks like the “common solution approach” à la JCOP never made it close to the HLT**
  - **Simple things: Histogram or output aggregation**

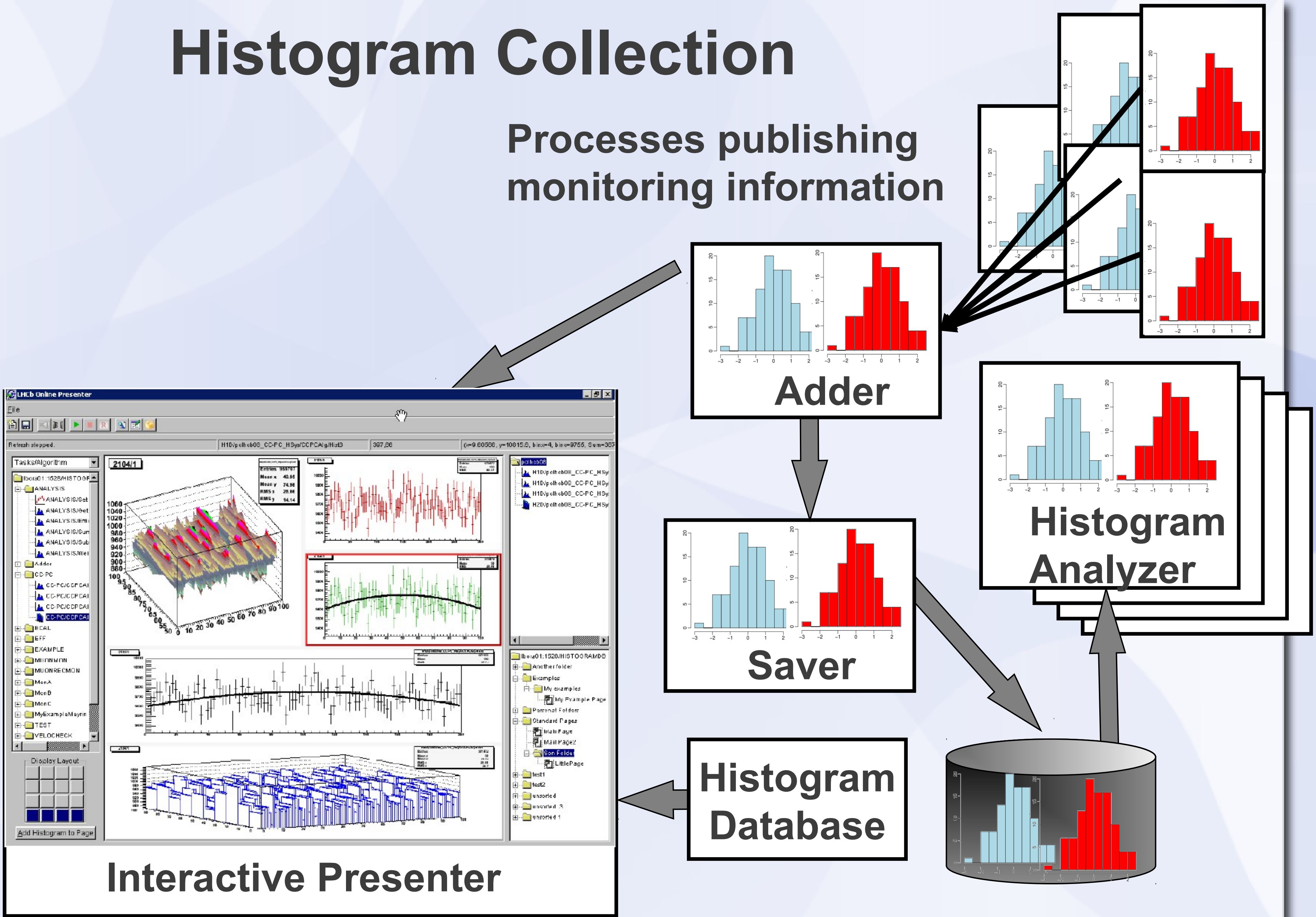
Backup Slides

Monitoring

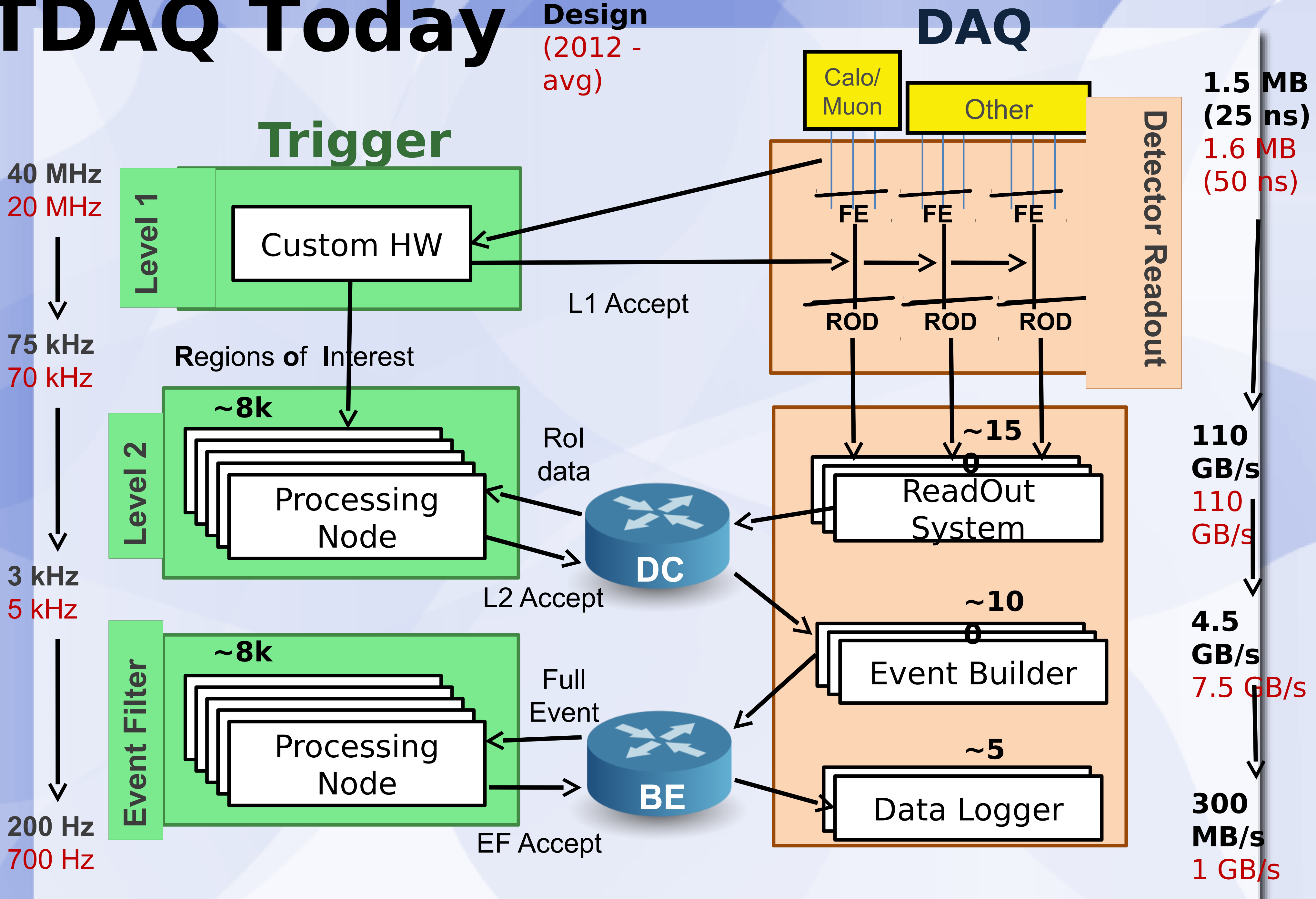


Readout Units
Event Data
Archive
HLT SubFarms X 56
Alarm UI
Status
Counters
Rates
Alarms
HLT
HLT SF Ctrl
Control System
UIs
Infrastr
Dataflow
DQHisto
(aggregation)
D I M
HLT Nodes X ~30
Automatic Actions
Detailed UIs
Detailed Counters
History Histos
Histogram Presenter
Histos Alarms
Histo Savesets
Histos
Automatic Analyses
Storage

Histogram Collection



TDAQ Today



Design
(2012 - avg)
DAQ
Trigger
40 MHz
20 MHz
75 kHz
70 kHz
3 kHz
5 kHz
200 Hz
700 Hz
Level 1
Custom HW
Regions of Interest
L1 Accept
Level 2
~8k
Processing Node
RoI data
L2 Accept
DC
Event Filter
~8k
Processing Node
Full Event
EF Accept
BE
Calo/ Muon
Other
Detector Readout
FE
FE
FE
ROD
ROD
ROD
~150
ReadOut System
~100
Event Builder
~5
Data Logger
1.5 MB (25 ns)
1.6 MB (50 ns)
110 GB/s
110 GB/s
4.5 GB/s
7.5 GB/s
300 MB/s
1 GB/s