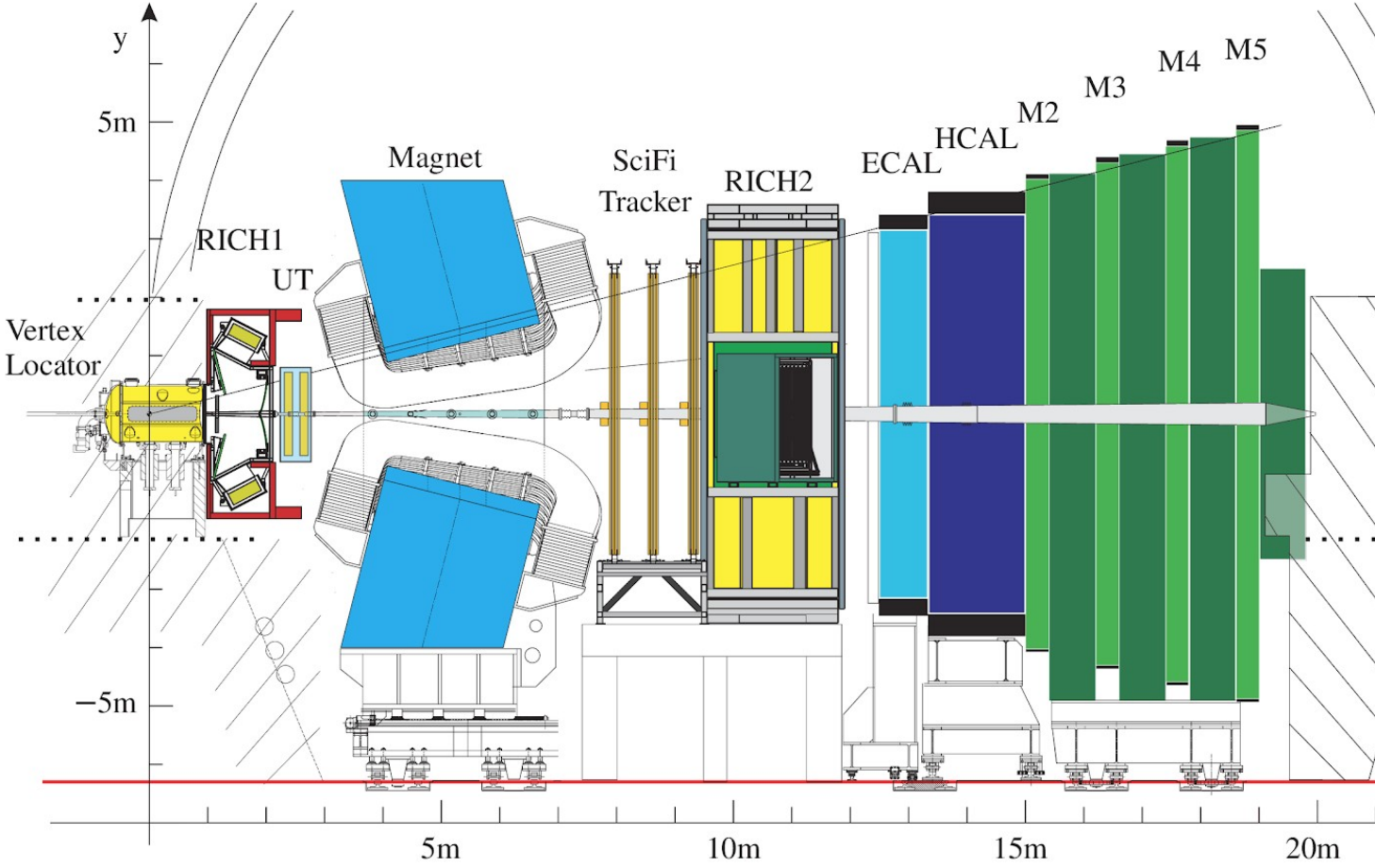
Streaming Readout X: LHCb DAQ System Overview and Experience

Based on slides by Tommaso Colombo
Niko Neufeld, CERN

Jefferson Lab, USA
(talk given remotely)
May 2022

LHCb Upgrade 1

- Single-arm forward spectrometer at the LHC
- p-p bunch crossing rate: 40 MHz (about 30 MHz colliding bunches)
- Luminosity: 2×10^{33} $cm^{-2}s^{-1}$



Trigger-less readout: why?

- With traditional calorimeter+muons trigger:

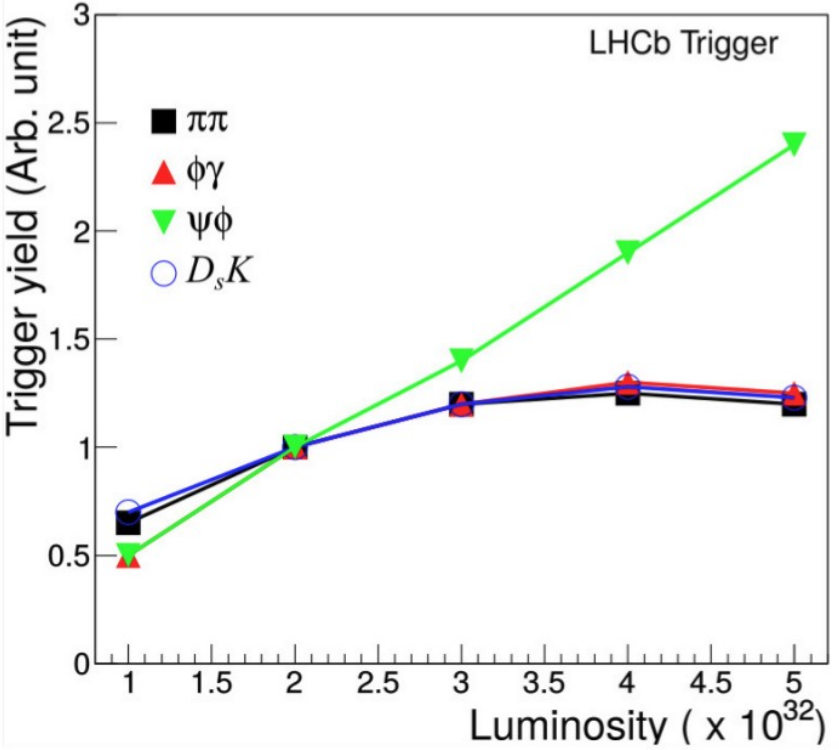
  Increase in luminosity

  ≠

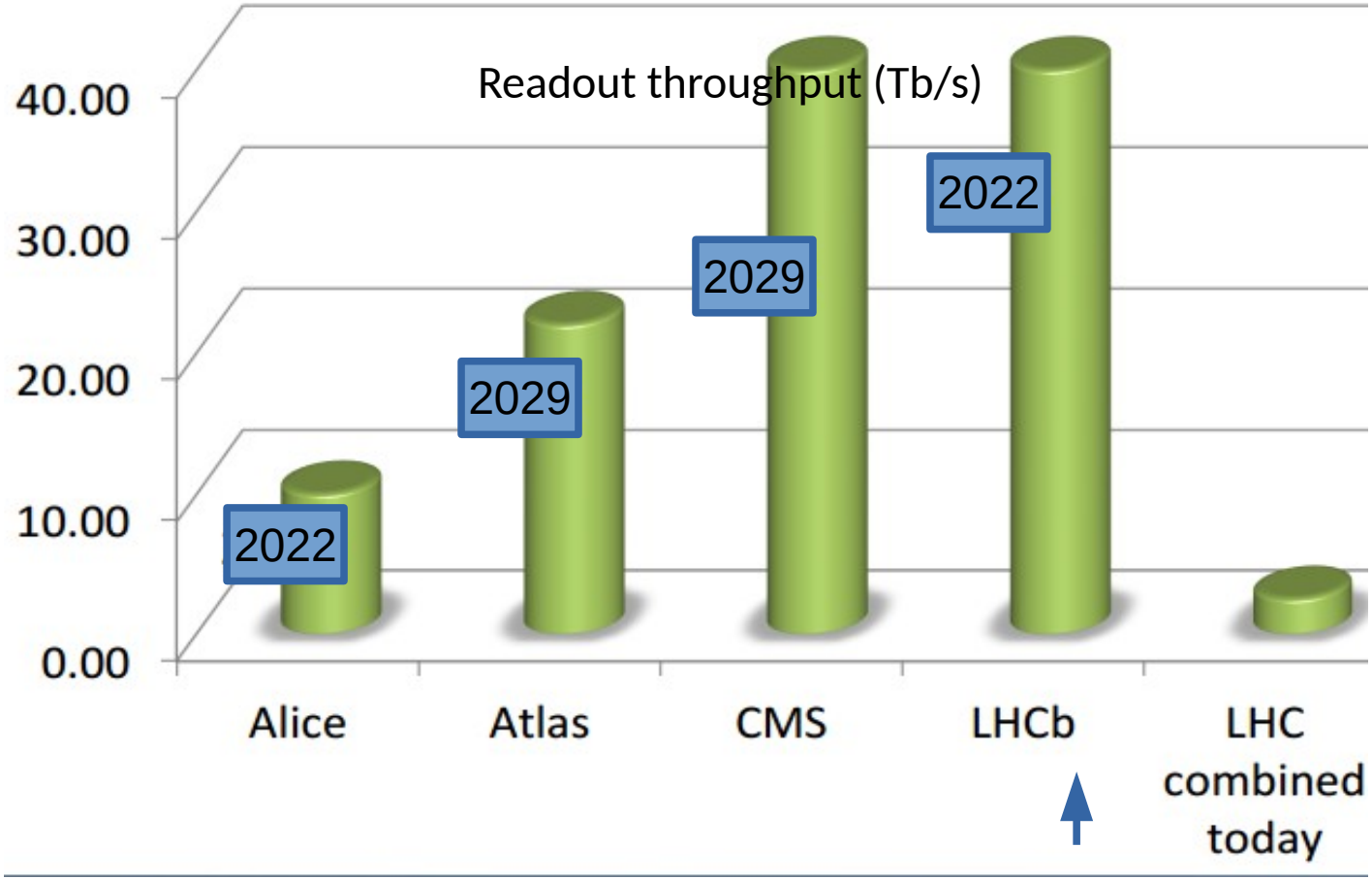
  increase in “interesting” events

- As luminosity grows, thresholds must be increased to keep rate constant
- Trigger inefficiency from higher thresholds is not compensated by higher lumi

Low level trigger yield vs Luminosity (cm^{-2} s^{-1}) for a trigger rate of 1 MHz



Trigger-less readout: when?

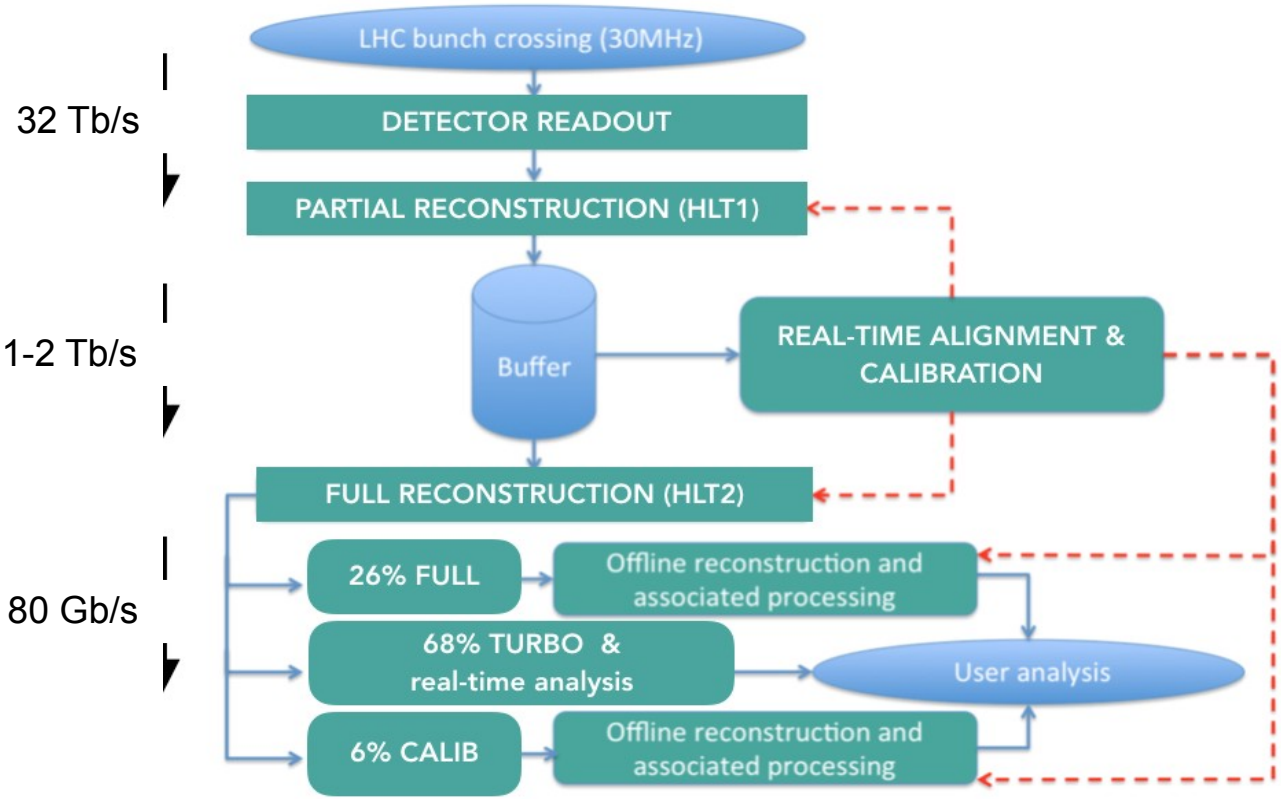


Trigger-less readout: how?

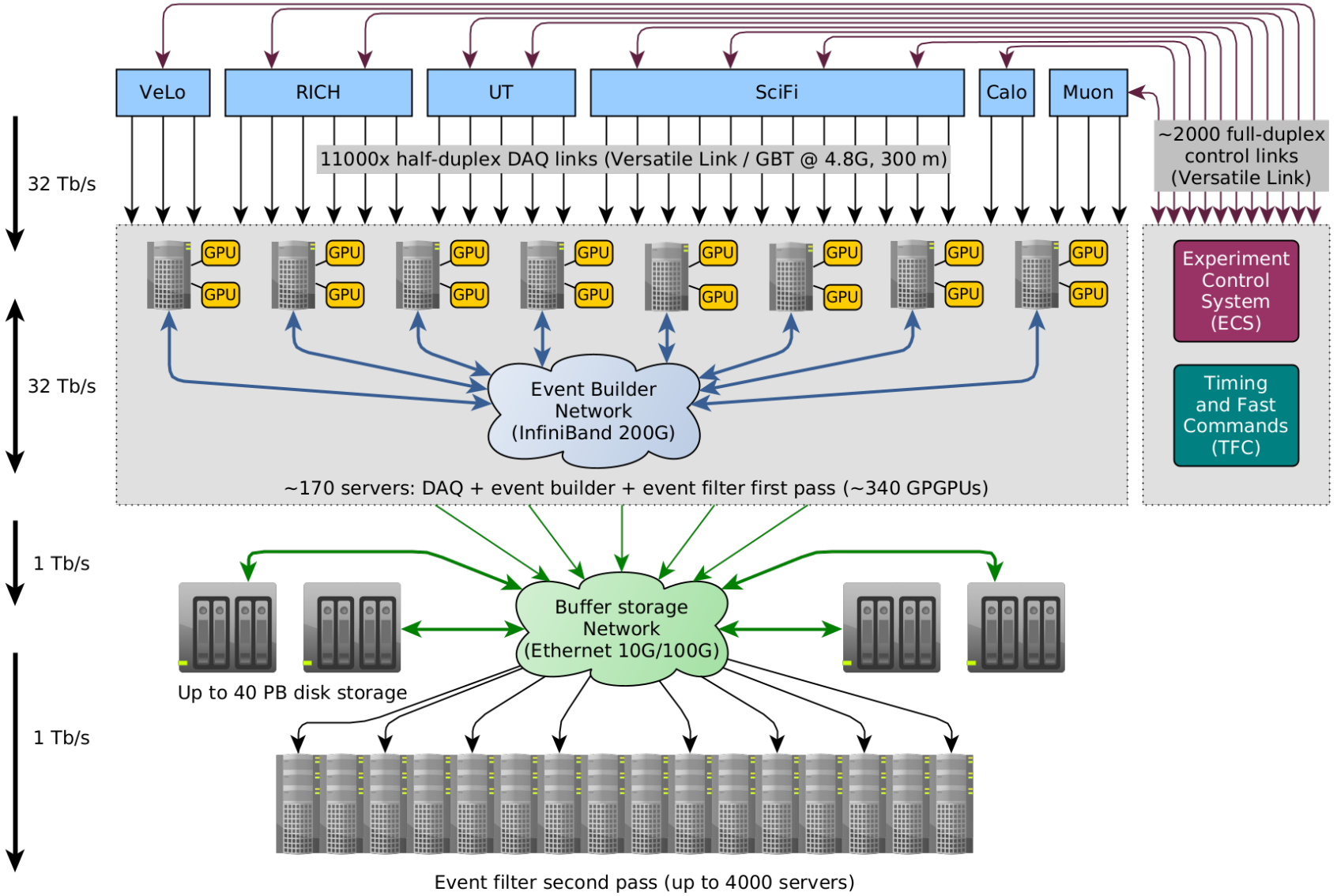
- Spectrometer geometry:
  fibres/cables are not "in the way"
- Relatively low radiation levels allow relaxed radiation-hardness requirements for FPGAs in many detector front-ends
- Zero-suppression on the detectors
- Total event size comparatively small (~100 kB)
- Bonus:
  software trigger can do online selection with offline-like reconstruction

Data-processing and selection

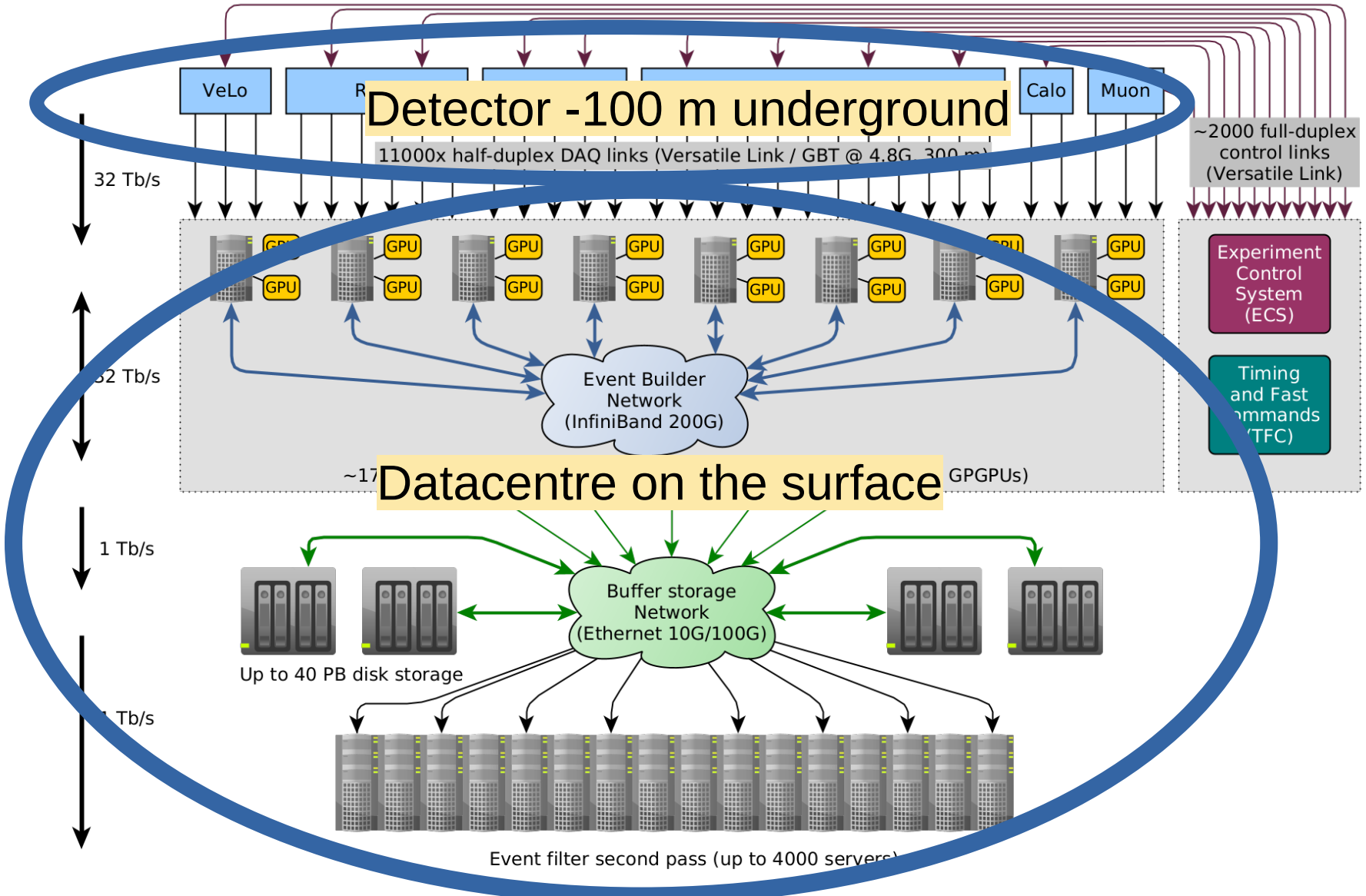
- Two stages of software filtering:
  1) "HLT1" on GPGPUs
  2) "HLT2" on CPUs
- Large storage buffer to decouple the two
- Calibration and alignment are performed "semi-live", while the data are buffered



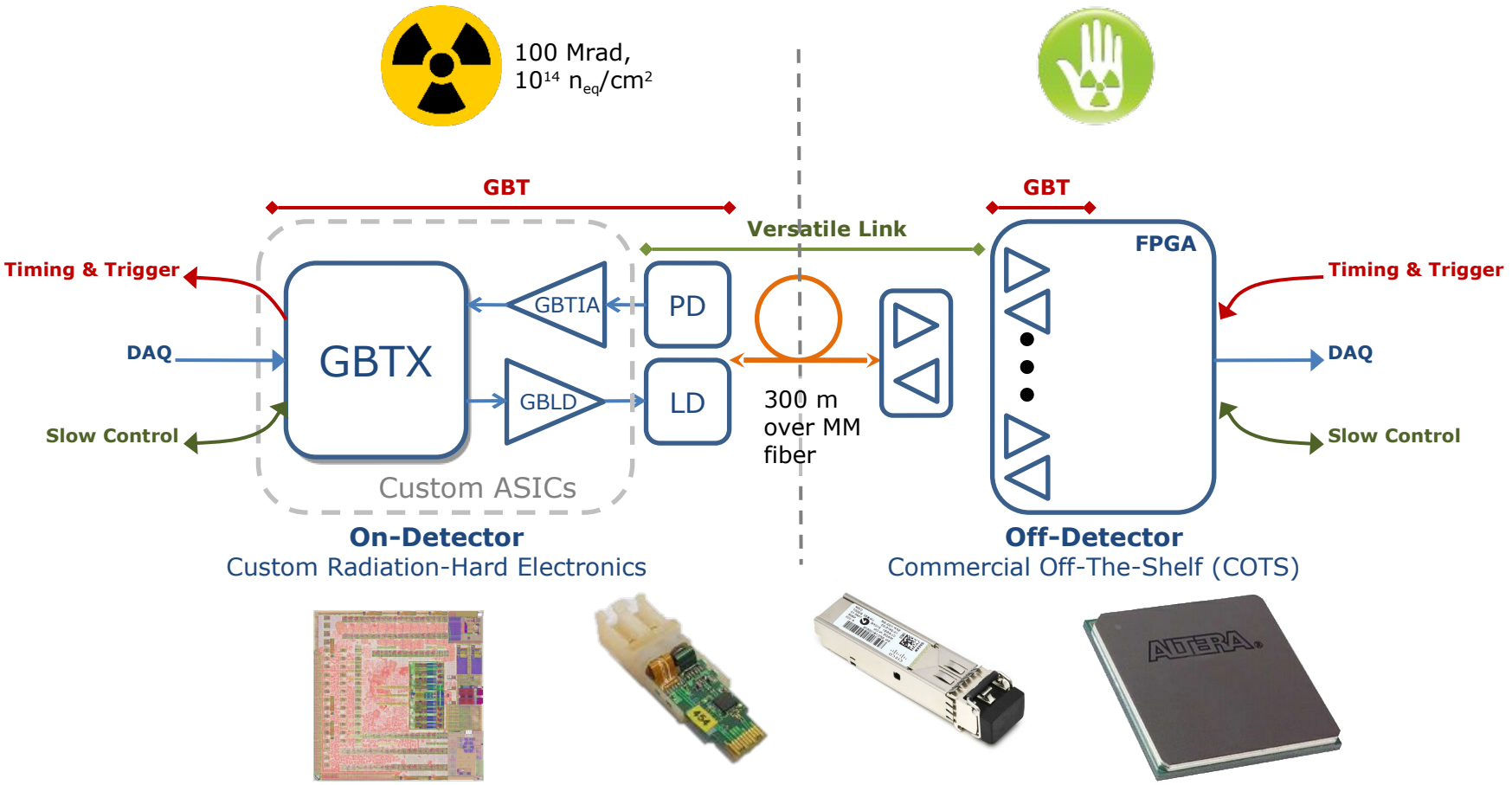
System overview



System overview



Example: 4.5 – 10 Gbit/s front-end GBT over Versatile Link

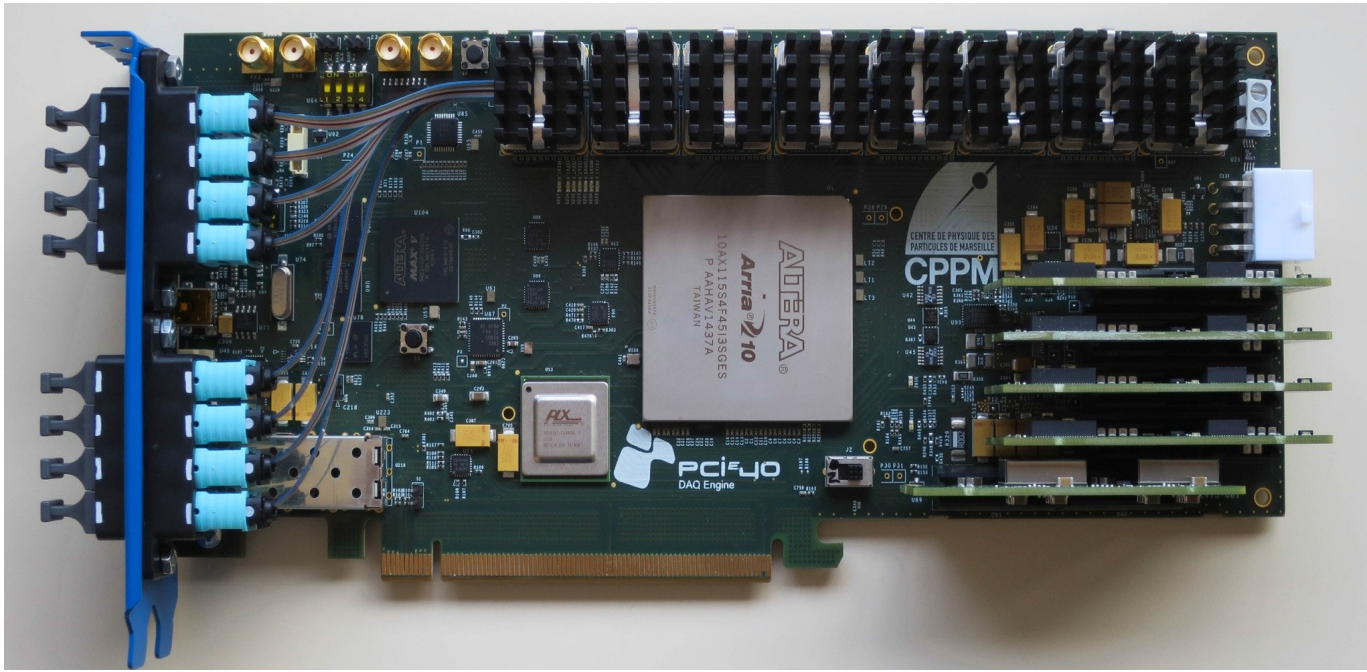


Credit:
P. Moreira
S. Baron
(CERN)

Back-end: PCIe40

A single custom-made FPGA board for DAQ and Control

- Based on Intel Arria10
- 48x10G-capable transceivers on 8xMPO for up to 48 full-duplex Versatile Links
- 2 dedicated 10G SFP+ for timing distribution
- 16x PCIe 3.0



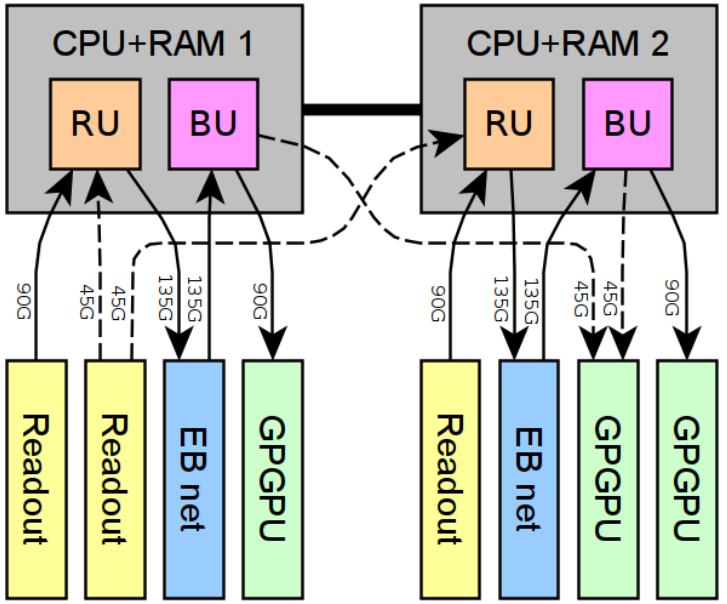
One board, many firmware personalities

478 Readout Boards (TELL40)

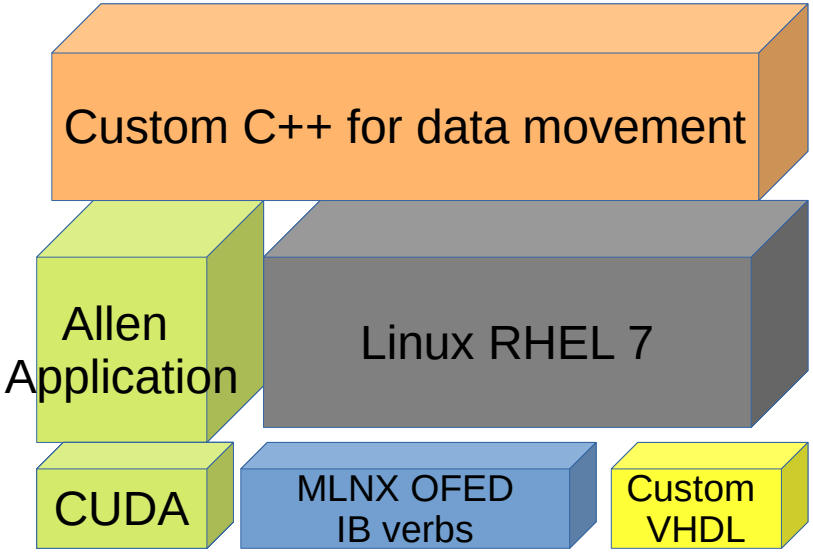
- Data Acquisition
- First pre-processing of the data
- E.g.:
  - Re-ordering and separation on event boundaries of streaming data
  - Hit clustering

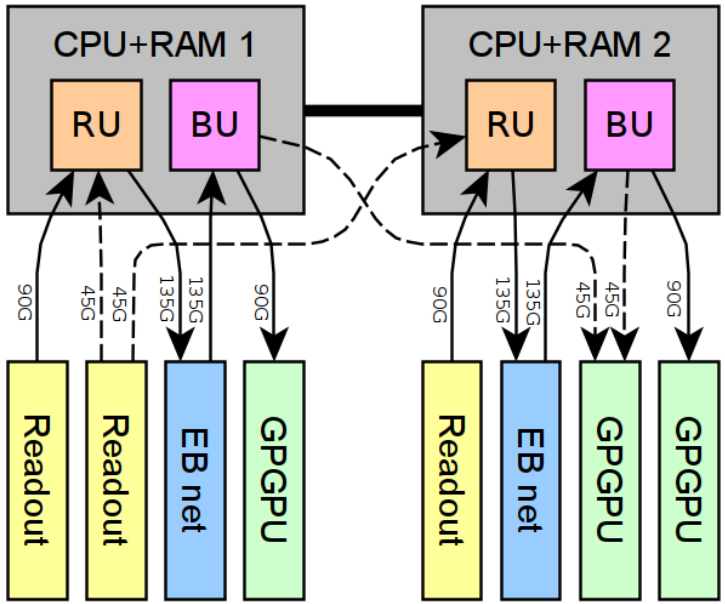
Event builder server

- 2 AMD EPYC 7002-series CPUs
  - PCIe 4.0
  - 8+8 DDR4 channels
- 3 readout boards
- 2 InfiniBand 200G NICs
- Up to 3 GPUs
- 512 GiB RAM (buffer to decouple 2 stages of data-flow)



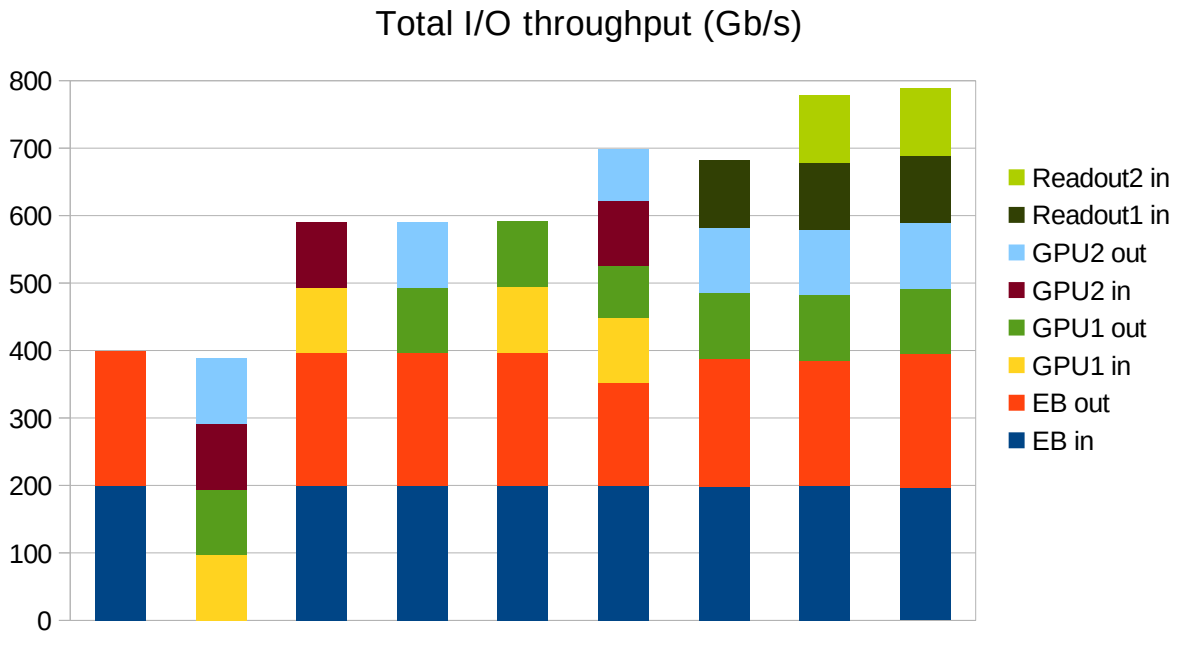
Software Stack

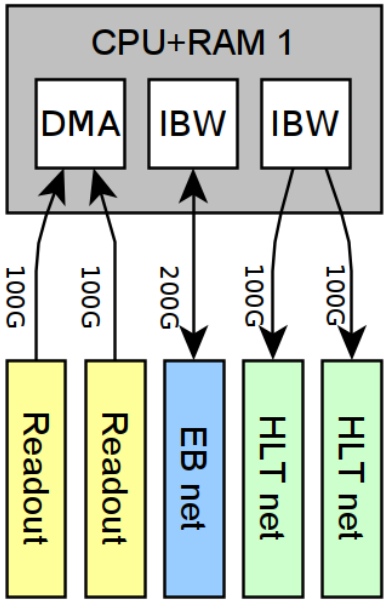




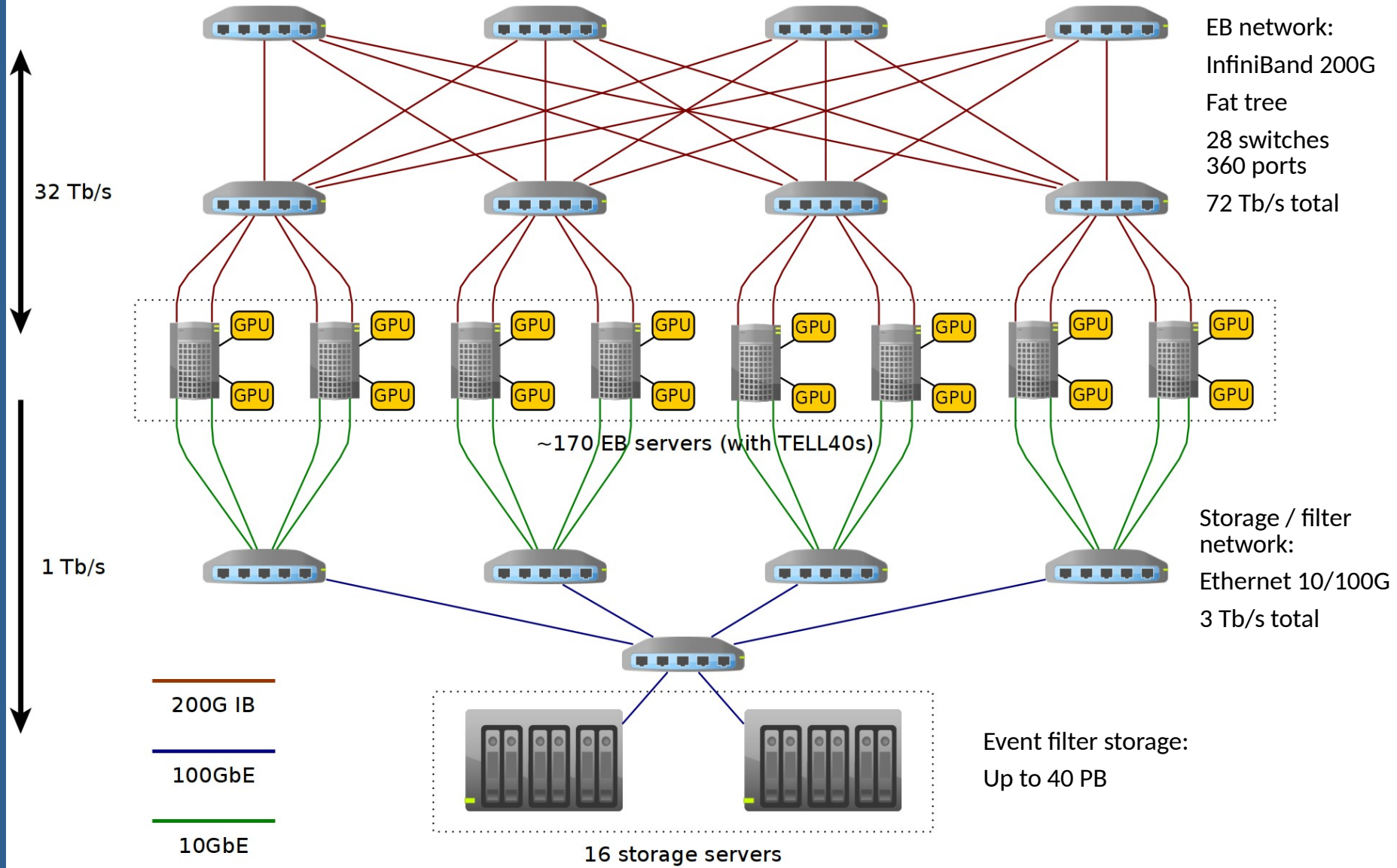
Challenges for EB servers

Memory subsystem pushed to the limits! RDMA is crucial.





Event builder networks



EB network:

InfiniBand 200G

Fat tree

28 switches
360 ports

72 Tb/s total

Storage / filter network:

Ethernet 10/100G

3 Tb/s total

Event filter storage:

Up to 40 PB

Challenges for the EB network

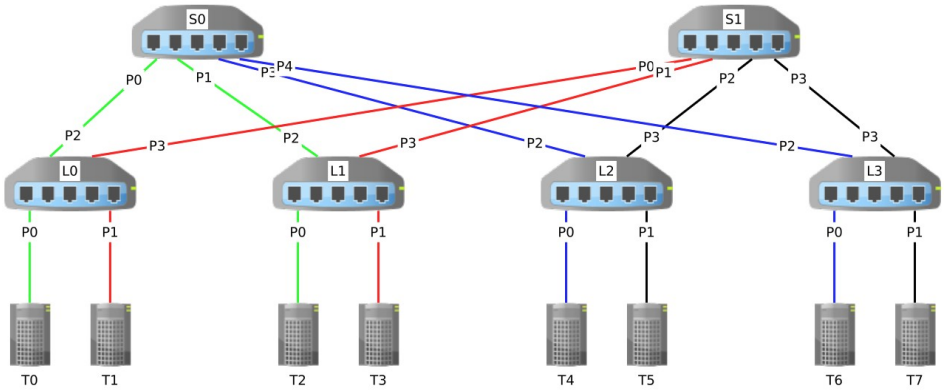
- Needs to collect data from 478 readout boards into a single "location"
- And hand it over to GPGPUs + CPUs for further processing
- Want high link-load (keeping costs low)
- Want to use some kind of remote DMA to reduce server-load
- Traffic is inherently congestion-inducing

→ Our solution: careful application-level traffic scheduling

→ Specialized routing algorithm for our network topology (fat tree)

Event building, a.k.a. MPI_Alltoall

- Traffic pattern is *all-to-all gather*:
  For each event, one “builder” server receives fragments from all servers
- Schedule: linear shift
  - With N servers, the transfer of N events is divided into N phases
  - In every phase each server exchanges data with only one server
- If the start of a phase is synchronized, and the network is non-blocking
  → no link conflicts!



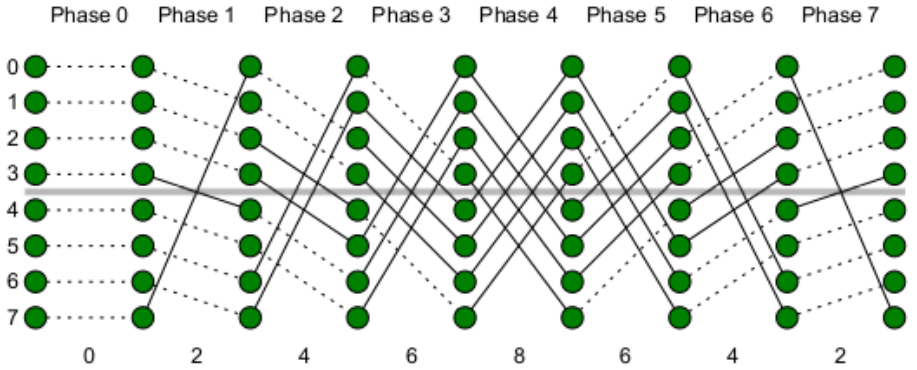
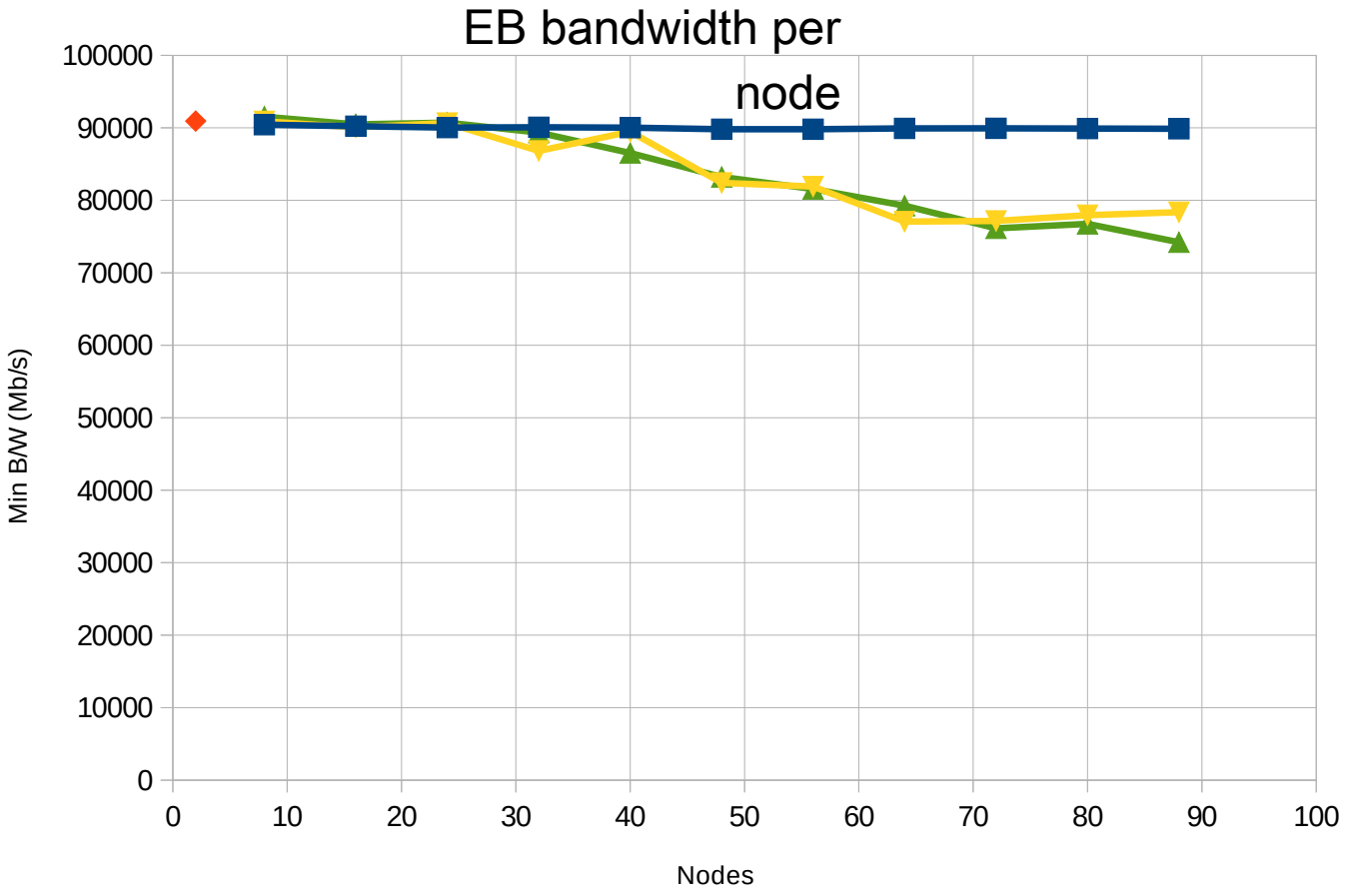


Image credit: B. Prisacari et al.

Scalability on InfiniBand



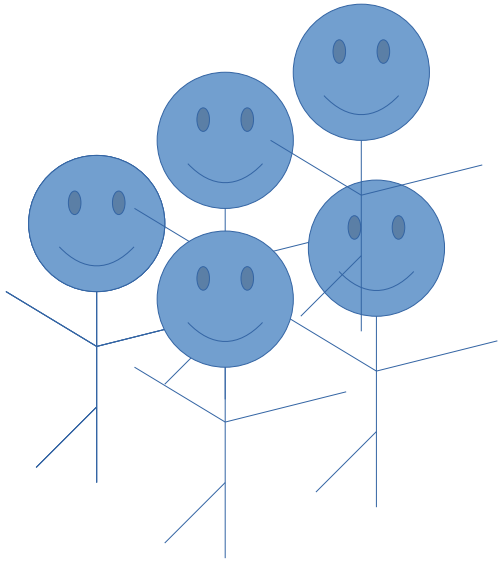
Tested at the Goethe-HLR HPC cluster (InfiniBand 100G)

With the right traffic shaping, almost

Why InfiniBand?

- Remote DMA is crucial for EB server performance:
  - RDMA implementations do not like packet drops:
    either deep buffers or good flow control are needed.
  - Deep buffers @ 100G = expensive.
  - Many Ethernet flow-control bugs found on available reference platforms.
- **Could never get access to a really big Ethernet test system:**
  Network congestion issues only appear at scale.
  For InfiniBand we have used super-computer sites.
- Lowest risk&cost solution – at technology decision early 2020 – is InfiniBand
  With additional effort & time, no doubt that also RoCEv2 can be made to work

Resources: Money & People



Resources

- Long distance optical fibres: 1000 kUSD (material + installation)
- Event-builder servers 1800 kUSD
- InfiniBand networking: cables, fibres, transceivers, network adapters, switches: ~ 800 kUSD
- Ethernet networking: cables, fibres, switches: 200 kUSD
- Storage: disks (51 PB raw) + front-end servers (16) + associated network ~ 1200 kUSD

- Five full time engineers for firmware, hardware and software of dataflow (no "physics")
  - Small number (2 – 4) students
- Supported by system administration team (4 people) and control system engineers (2 full-time)

Experience so far: What didn't work as expected

- **FPGA firmware**: test-coverage not complete, some issues only seen at scale and only in the real system, even for firmware working perfectly well in simulation and on integration test hardware (for weeks)
- **Fibre infrastructure**:
  - All fibres were installed and tested by professional company (20000 test-reports delivered)
  - Yet: 2 years after installation observe a small amount of broken (< 1%) (at the patch-panel) fibres. There are spares but in many cases the cabling plan needs to be completely changed → very laborious (relabelling, rerouting, etc...)
- EB **server nodes**: the most cost effective solution (in our analysis) uses GPU servers with 8 PCIe Gen4 slots. We have tested these systems with the vendor since the engineering sample and everything looked good
  - Unfortunately in production we see a lot of issues related to the PCIe interfaces, not yet fully solved.

Experience so far: What I would do differently the next time

- In this DAQ one pays for two things: PCIe slots and network ports
- Our system is made to minimize both, leading to a “folded” event-builder where every node is sender and receiver
  - Tight constraints on number of GPGPUs / accelerators, readout-cards and NICs
  - Expensive servers, little choice
- “Unfold” the event-builder with dedicated senders and receivers
  - Doubles number of switch ports (but those are relatively cheap)
- InfiniBand is great, Ethernet at about the same cost would be still better
  - Interoperability, common large network, same tools, etc...
  - Need to get RoCE (RDMA over Ethernet) to work
  - Could send directly from FPGA card

Summary

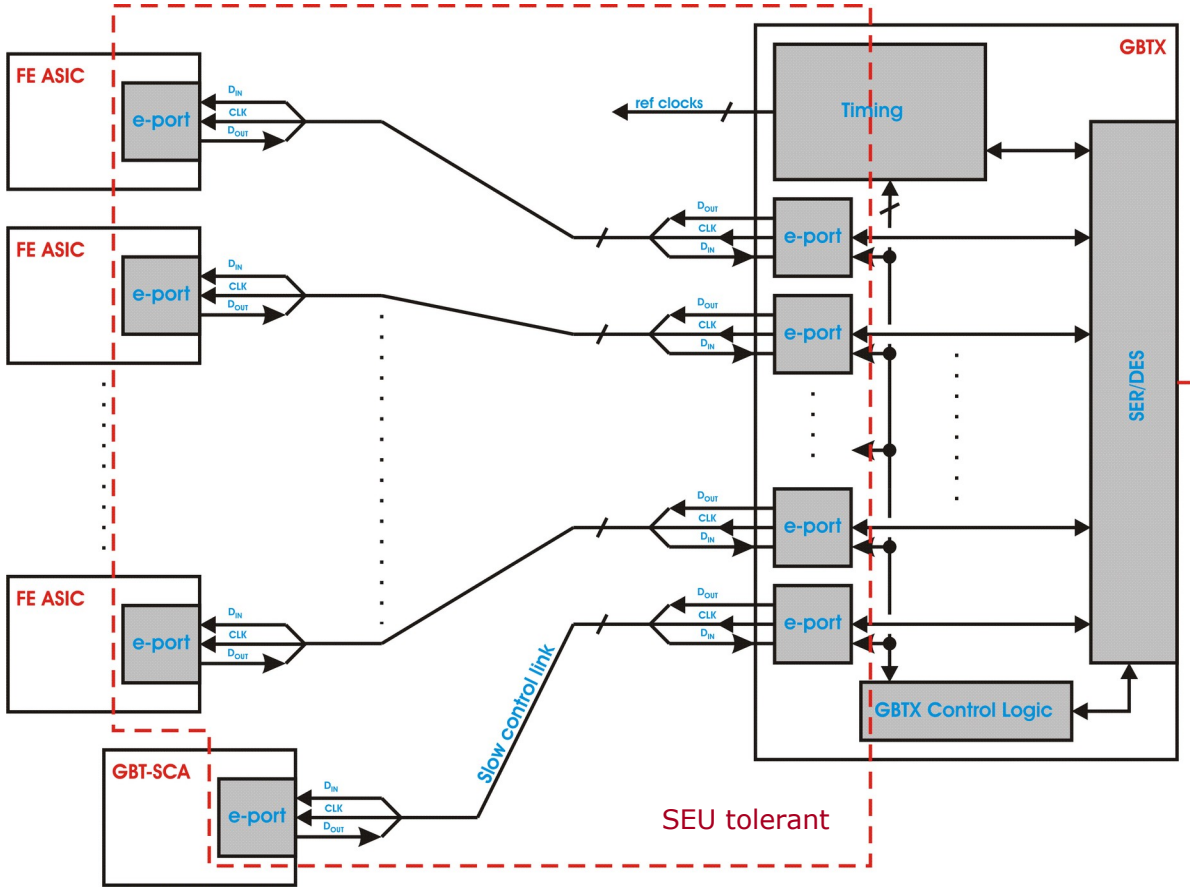
- LHCb can do and afford a full read-out at bunch-crossing rate
- Single stage synchronous readout built around a single flexible FPGA board
- AMD Rome (PCIe Gen4) based servers make compact,
  very-high-I/O event-builder, connected with 200 Gb/s InfiniBand
- Event-selection is entirely in software to maximize physics yield, increase the amount of data collected, flexibility and minimize cost
- The system is very well scalable, by up to 3 a factor without any substantial changes

Further improvements & R&D

- “In-flight” processing, by processing on CPU/GPU while receiving /transmitting data (independent of host) (a la “Bluefield”)
  - Particularly interesting if data-reduction can be achieved to save memory and/or network bandwidth
- Direct transfer on PCIe / CLX –> save memory bandwidth in host

Additional Material

Front-end: GBTx multiplexing

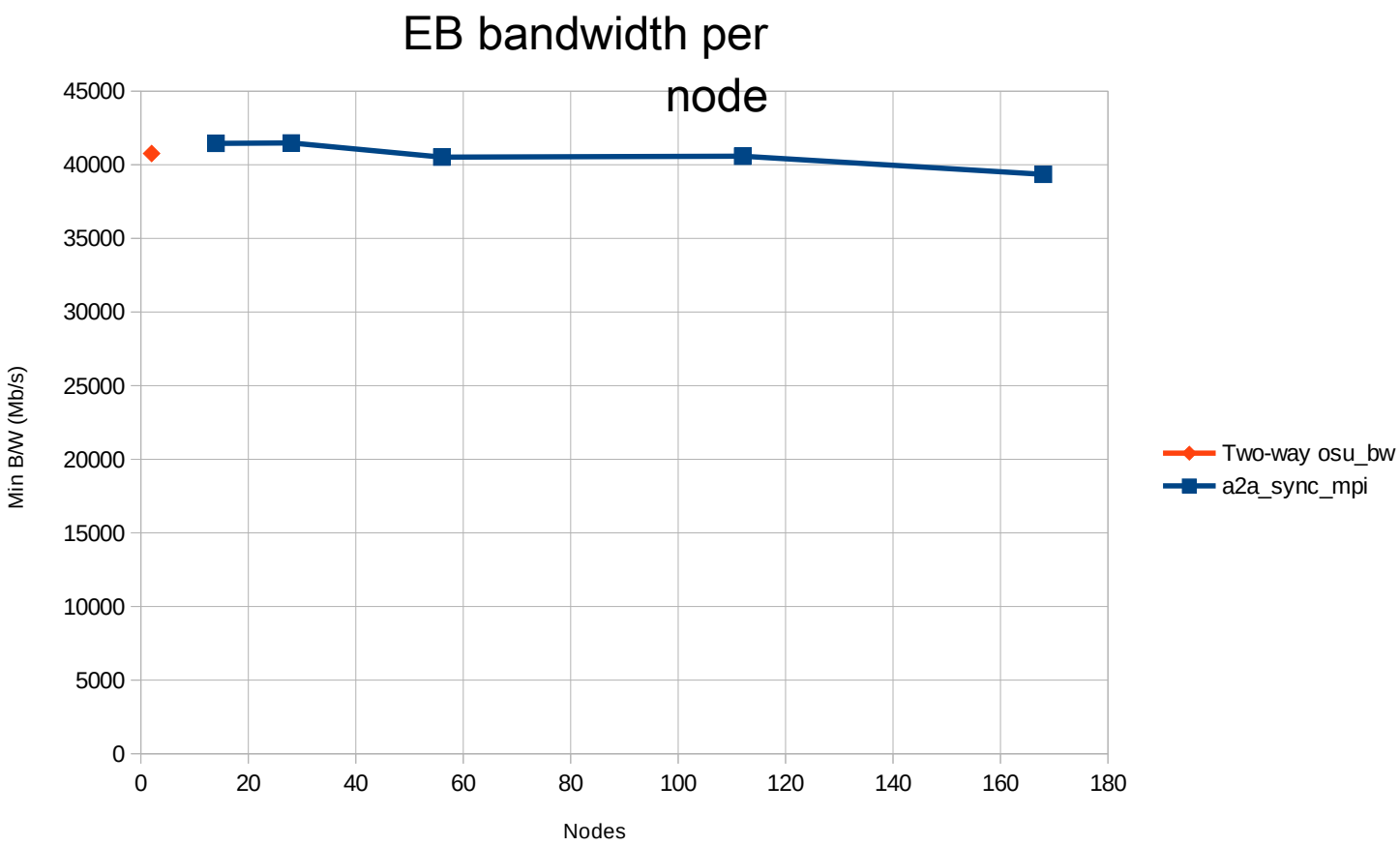


- GBT/Frontend interface: Electrical links (e-link)
  - Serial, bidirectional
- Up to 40 links per ASIC
- Programmable data rate:

  40×80, 20×160,
  or 10×320 Mb/s

Credit:
P. Moreira
(CERN)

Scalability on InfiniBand

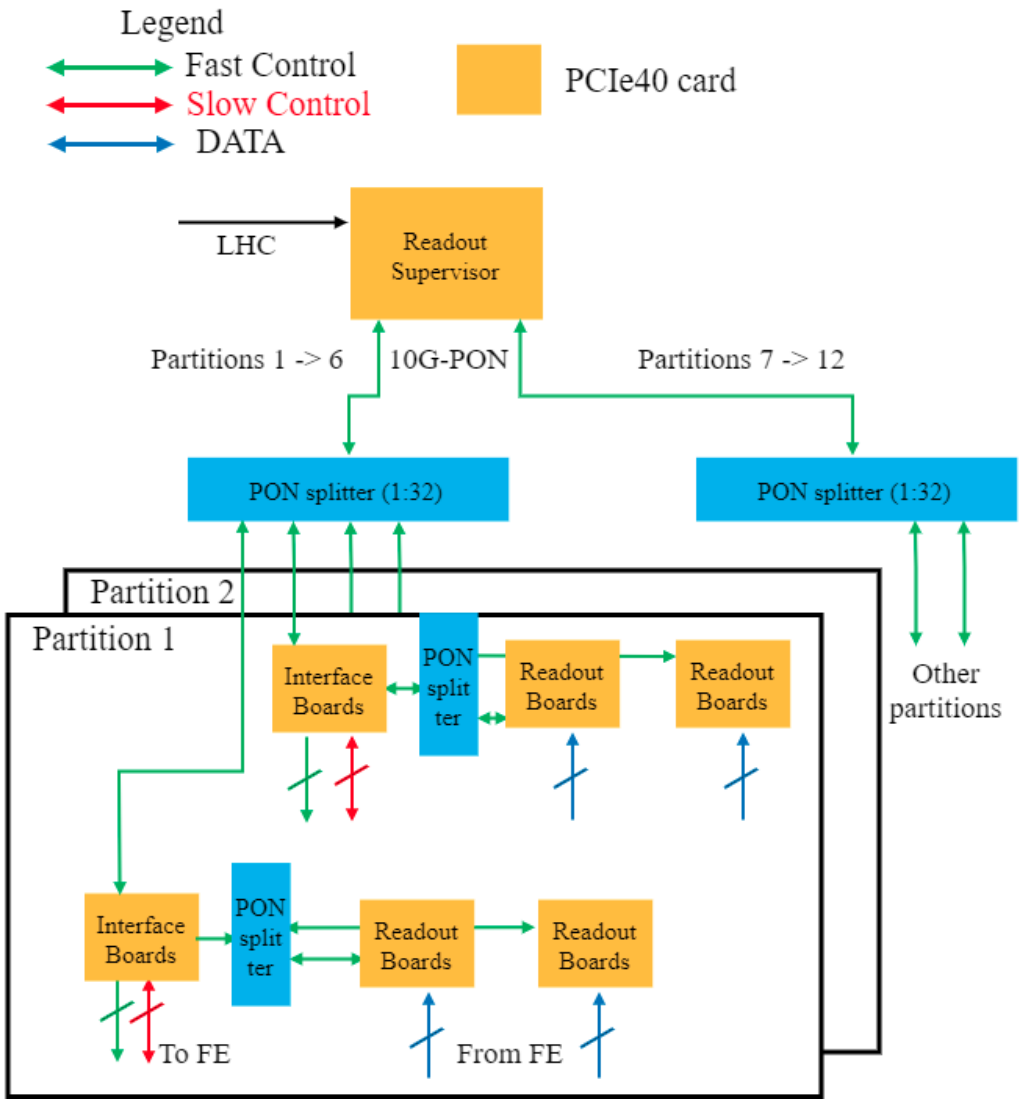


Tested on the CMS DAQ (InfiniBand 56G)

Very good scalability with almost 200 nodes

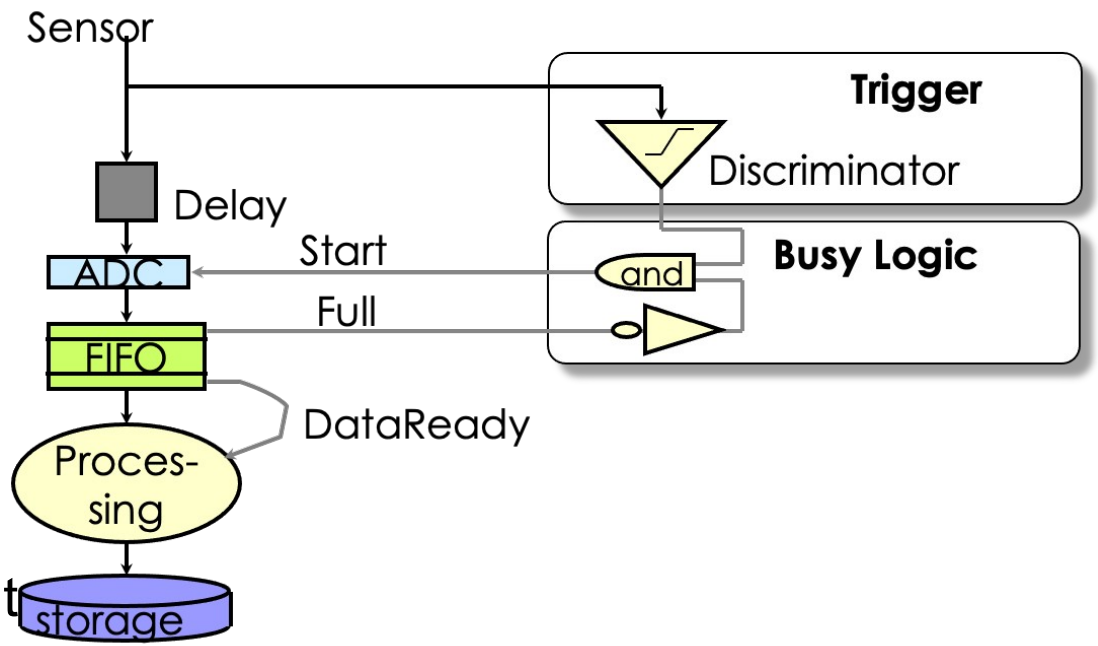
Timing and Fast Commands

- Synchronously driving the Front-End electronics over GBT
- 10G-PON for efficient Back-End signal distribution and fixed phase clock recovery
- Partitioning for debugging and commissioning



Triggered readout for high rate experiments

- Suppose you have a million channels sampled at 10 MHz
- A typical approach would be like in the opposite drawing
- There are many variations of this scheme
- The trigger is crucial in that it limits the rate at which data are digitized and put into the readout FIFO
- The “Delay” in practice will be some kind of analogue buffer



- The “Delay” in practice will be some kind of analogue buffer

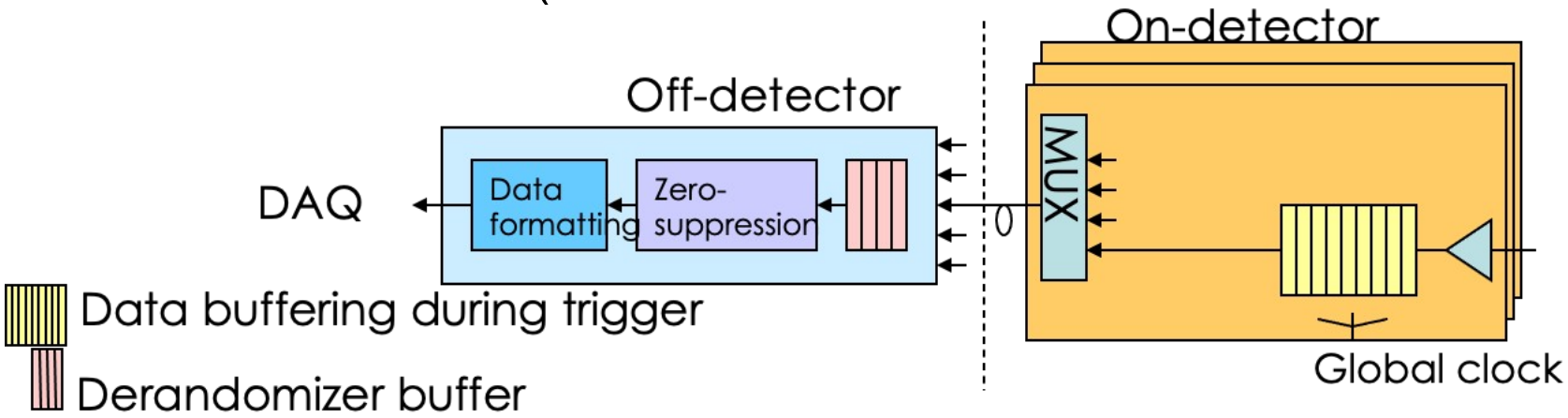
Disadvantages of a trigger

- Hard real-time logic is introduced → very difficult to use general purpose compute (CPU, GPU)
- A buffer is needed with a local selection mechanism → complexity on the front-end
- Often there is a (painful) compromise necessary between cost, power-consumption, complexity and selectivity
- Custom-trigger logic is often not easy to adapt or to maintain
- In experiments with many channels a trigger is only really “saving” something if it can work with a (small) subset of the total data, otherwise one must solve the problem, to be avoided in the first place (i.e. the “full readout at high rate”)
- Specifically for hadron colliders: radiation tends to exacerbate many of these problems!

To be sure: all of these can be overcome – at a cost, and not all apply to all experiments.

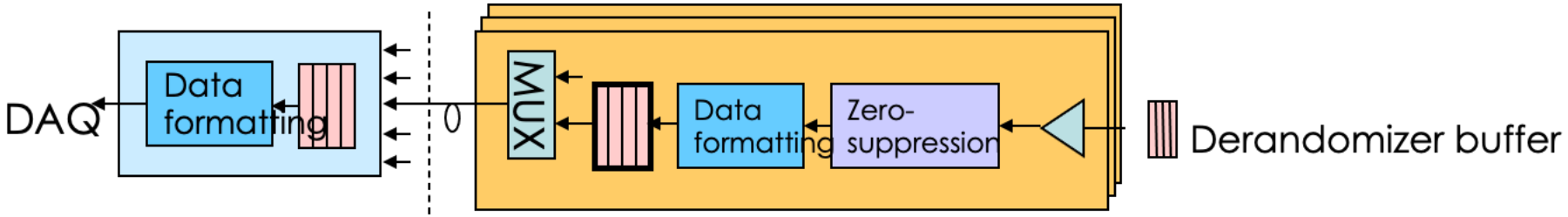
Removing the trigger

- Great simplification of the front-end (shown for the synchronous case, asynchronous would use some local clock)
- Needs a large number of (high band-width) links between front-end / on-detector and back-end / off-detector
- A lot of zeroes are sent :-(



Reducing the problem: 0-supp in the front-end

- Reintroduces some complexity to the front-end (but offloads the back-end from this task)
- For high granularity detectors can greatly reduce the number of DAQ links



An Example

- The LHCb read-out for LHC Run3
- Trigger-free, single-stage read-out
- How is it made?
- What does it cost :-)?
- What does one gain?