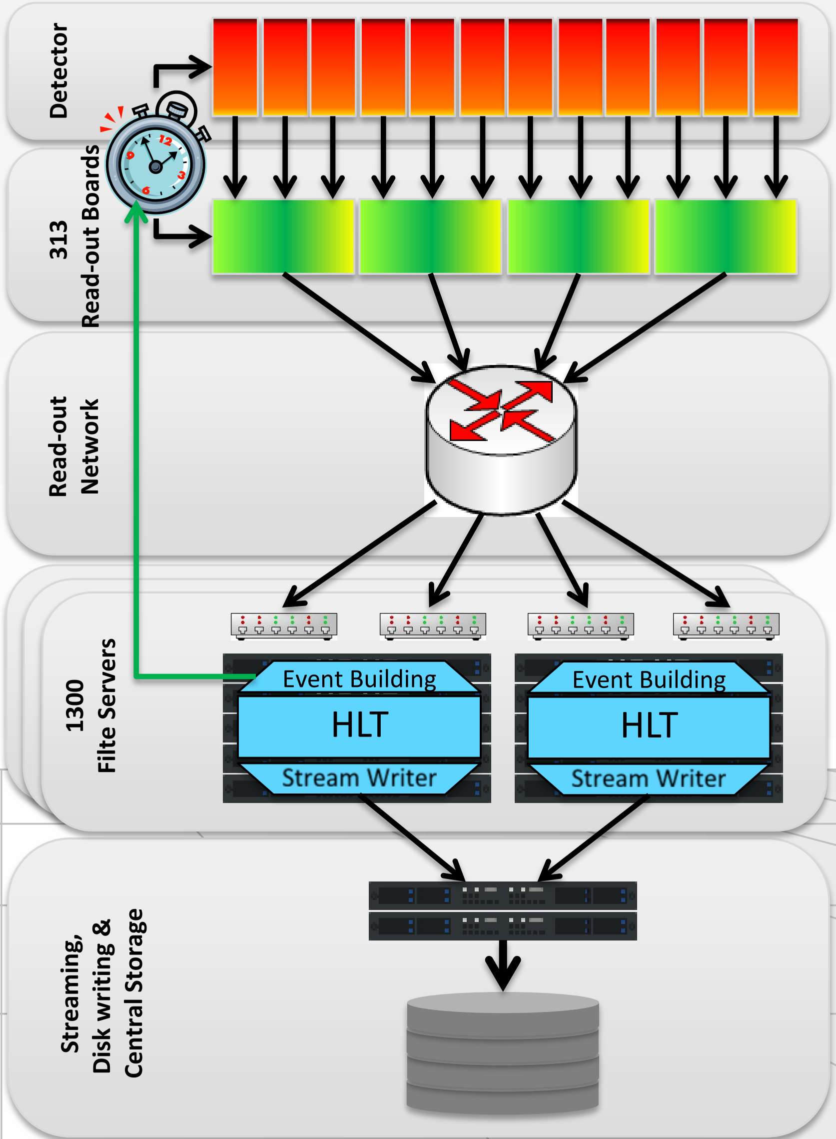
The LHCb Data Acquisition during LHC Run 1

F. Alessio, L. Brarda, E. Bonaccorsi, D.H. Campora Perez, M. Chebbi, M. Frank,
C. Gaspar, L. Granado Cardoso ,C. Haen, E. v. Herwijnen, R. Jacobsson, B. Jost,
N. Neufeld, R. Schwemmer, Vijay Kartik, A. Zvyagin,

CHEP 2013

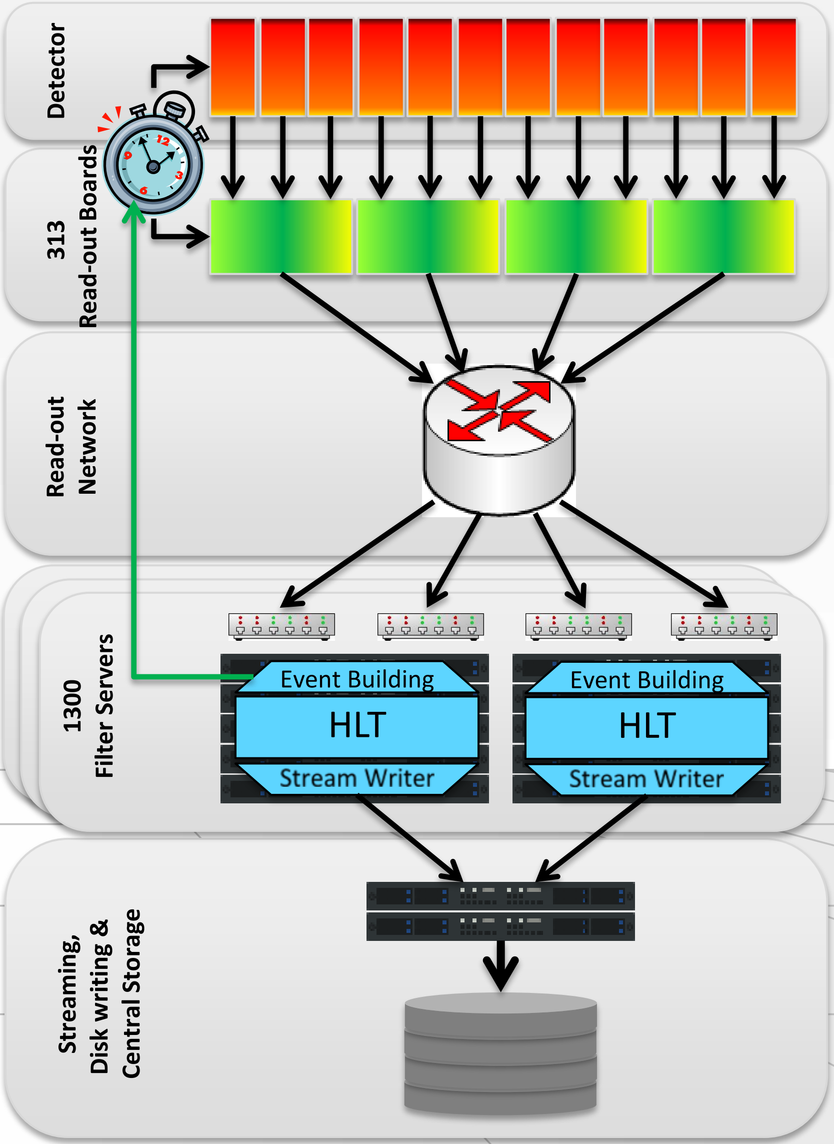
From Front-End to Hard Disk

- $O(10^6)$ Front-end channels
- 300 Read-out Boards with 4 x 1 Gbit/s network links
- 1 Gbit/s based Read-out network
- 1500 Farm PCs
- >5000 UTP Cat 6 links
- 1 MHz read-out rate
- Data is pushed to the Event Building layer. There is no re-send in case of loss
- Credit based load balancing and throttling



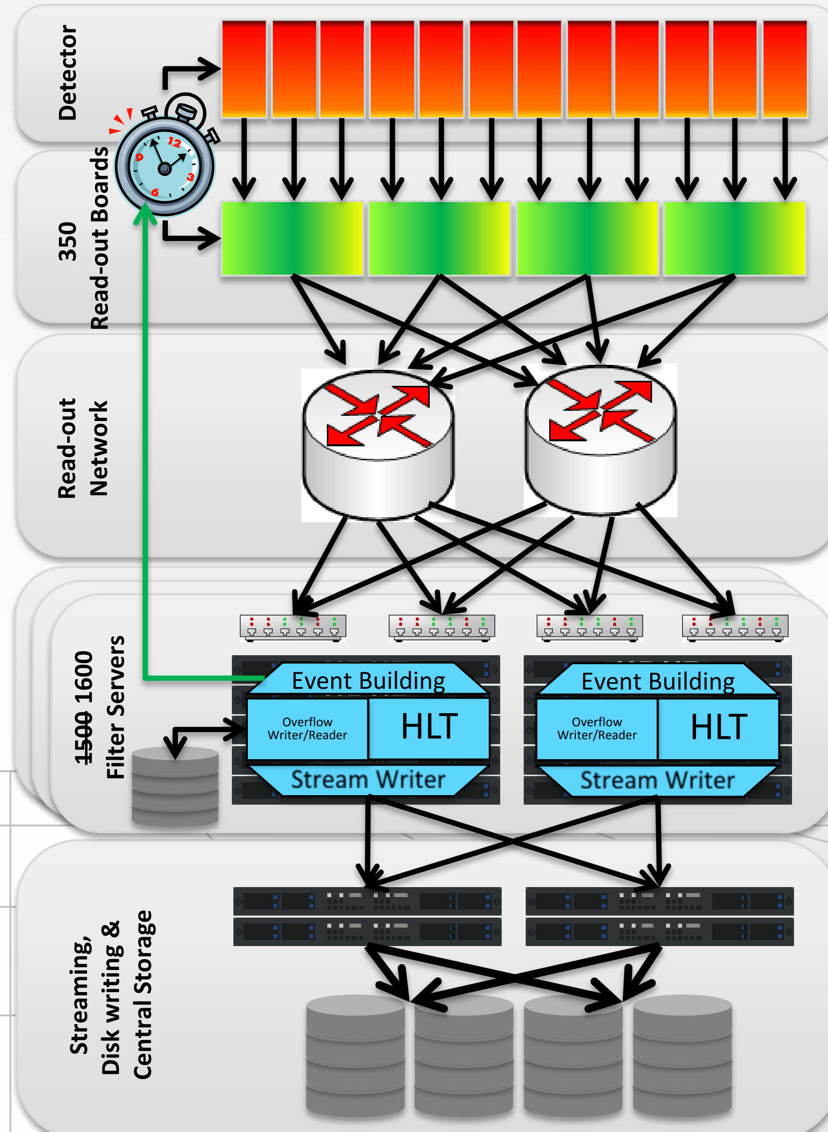
A bit of history

- Original DAQ Specs
  - Readout rate: 1 MHz
  - Up to 16 consecutive triggers
  - Total event size: 35 kB
  - HLT output rate: 2000 Hz
  - HLT output bandwidth: 80 MB/s
- Original DAQ architecture
  - More or less what's on the right
  - Single Core router
  - Data that can't be accepted by the HLT is throttled away.
- Original Storage Back-end
  - Controls software is served from central NFS/Samba servers
  - Trigger software is served from central NFS servers (Diskless Farm)
  - Monolithic Disk array
  - Good redundancy in data writers
  - Weak redundancy in File Systems and NFS/Samba servers



Where are we now?

- Current DAQ Specs
  - Readout rate: 1 MHz
  - Up to 16 consecutive triggers (sort of)
  - Total event size: ~~35~~ 50+ kB
  - HLT output rate: ~~2000~~ 5000 Hz
  - HLT output bandwidth: ~~80~~ 250 MB/s
  - 1000+ MB/s for special calibration runs
- Current DAQ architecture
  - ~~Single~~ Dual Core routers
  - Data that can't be accepted by the HLT is temporarily stored on HLT node for later processing
- Current Storage Back-end
  - Controls software is served from central NFS/Samba servers
  - Trigger software is served from central servers, but cached locally on farm node
  - Monolithic Disk array → internal separation between DAQ data and Software
  - Good redundancy in data writers
  - Self made NAS with OSS based High Availability NFS and Samba servers



Physical Installation (Core Router)

How did we get there (I)

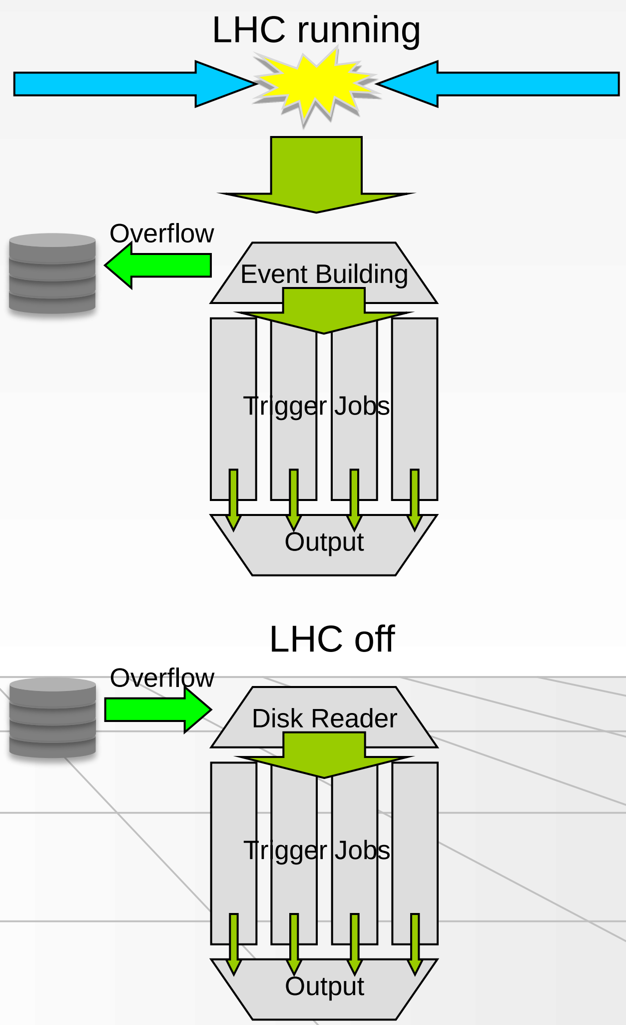
- Increase in Event Size:
  - Luminosity increase. Detector is running at twice its original design specs.
  - Design: $2x10^{32}$, Running at: $4x10^{32}$
  - Higher μ due to 50 ns bunch spacing → Higher detector occupancy → accepted events are bigger
- Output rate increase due to extension of physics program
- Calibration runs use up valuable beam time
  - The more throughput we have, the less beam time we lose
  - Can maintain more than 20 kHz of output with more than 1 GB/s of data rate for several hours
- VDM scans and SMOG runs for beam profile and luminosity measurements demand high throughputs
- Luminosity leveling
  - LHCb does not run at full LHC instantaneous luminosity
  - By continuously adjusting beams we do not suffer beam depletion over time
  - We have to store more data than anticipated per fill → More disks mean more throughput!

How did we get there (II)

- Moore's law helped a lot
  - Original estimates for computing power were very conservative
  - In fact we dropped another level of hardware triggers for software based triggering even before the experiment went online
  - Both trigger stages are run in software now
- Buy computing time through disk space
  - LHC duty cycle is not 100%
  - Process only a fraction of the incoming data immediately
  - Process the other fraction during inter fill gaps and technical stops

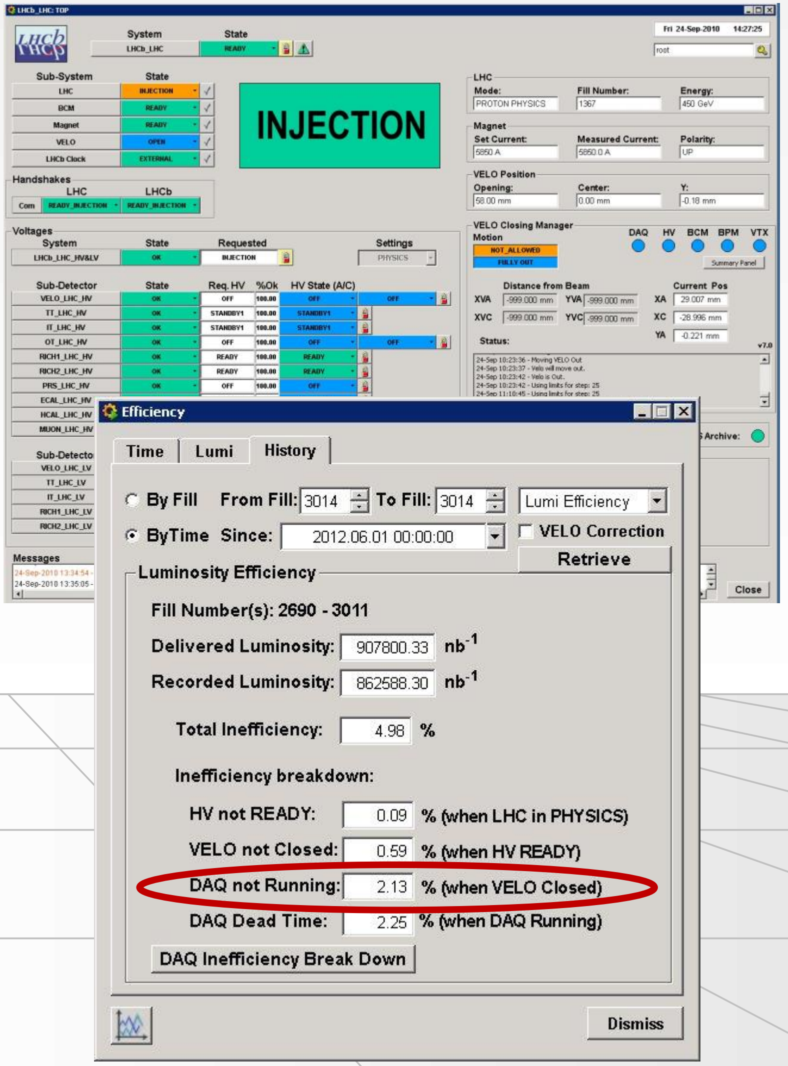
Deferred Trigger

- Procedure:
  - Equipped 1000 machines in our farm with 1-2 TB disks each
  - Over commit Farm by typically 30% while LHC is running
  - Data that can not be processed by the HLT node is written to the local disk
  - Once beams are dumped we start processing the data that has been temporarily stored on the disk
  - Make sure you don't process the same event twice!
- Side effects:
  - Data is not contiguous any more
  - Events of several runs can be in the system simultaneously
  - Disks like to fail, especially if there are many
  - It can sometimes take days before a fill has been completely worked off
  - Had to severely altered the design of the data flow and book keeping
  - The DAQ is constantly pumping out data
  - Less and smaller maintenance windows
- On the bright side:
  - Every failed farm node meant a reduction in DAQ performance
  - Now: other nodes just write a little more to disk



Automation

- Significant amount of efficiency is lost due to human latency
- Big Brother
  - Acts on state changes and communicates with the LHC
  - Automates common task sequences
    - Ramping of HV systems
    - Opening/Closing of the Vertex Detector
    - Calibration runs at EoF
  - Generally asks before acting
- Auto Pilot
  - Starts the system and keeps it running
  - Automatic recovery of common failures
    - Front-End recovery
    - Recovery of failed trigger jobs
    - Recovery of failed farm nodes
- Human intervention still necessary for unknown problems
- Speech synthesis program notifies operator in case of trouble
- → 98% DAQ efficiency since adoption



What went wrong?

Things that did not work quite as expected – Software Rollout

- HLT software roll-out on farm
  - Event Filter Farm is based on diskless nodes
  - Operating system and all software comes via NFS servers
- Software is very modular and organized in small, shared object libraries
  - Libraries are distributed over a large directory tree
  - Several versions of those libraries exist and the correct version is chosen by adding its directory to LD_LIBRARY_PATH of a job
  - LD_LIBRARY_PATH contains O(100) entries
- Launching an HLT job causes several thousand cache misses while searching for a particular .so file
  - No multi-threading → Multiple jobs per farm node
  - Cache misses are propagated to the NFS server
  - 17.000 Jobs are starting at the same time

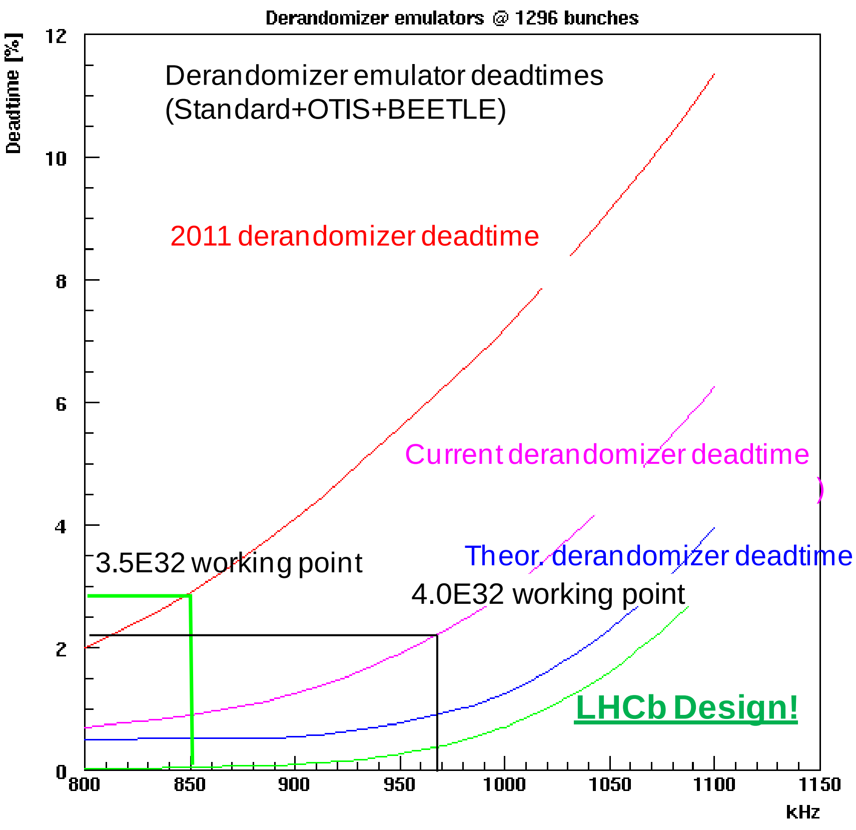
– Even with several very powerful NFS servers it would have eventually taken an hour to successfully launch all jobs

Things that did not work quite as expected – Software Rollout

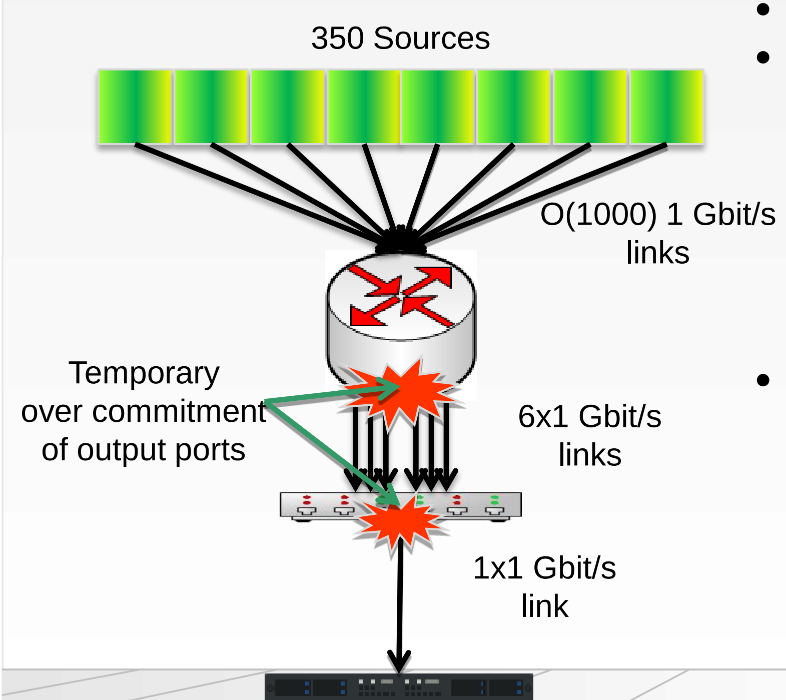
- First solution: Custom, Union-FS like Fuse File System
  - Copies all necessary files into local ram-disk on first application launch
  - Memorizes cache misses locally
  - Successive launches are served from ram-disk and internal DEntry cache
  - Slight disadvantage: Read only! Changes of directory structure are not allowed after directory has been seen for the first time
  - → Startup time: 5-6 mins
- Second step: Launch less jobs
  - Each machine starts only 1 job and clones are *fork()*ed once job has been fully configured and is running
  - Additional benefit: Reduces memory consumption by sharing static pages
  - → Startup time: 2 mins
- Last step:
  - Create memory checkpoint image of running job and store it as monolithic file
  - File is distributed to farm nodes via Bit Torrent protocol
  - Launch single job and fork clones once fully configured
  - → Startup time: O(seconds)

Things that did not work quite as expected – Dead time less read-out

- TDR Specs:
  - 1 MHz L0 rate
  - 16 consecutive triggers
- Front-ends (mostly) fulfill the specs
- What happens after the 16 consecutive triggers?
  - Some FEs need more time to recover from trigger trains due to organization of internal buffers
  - Some FEs have problems with certain trigger patterns
- VHDL code of FEs integrated into Read-out Supervisor FPGA
  - Internally emulates Front-end
  - Determines when buffers would overflow
- While we can't fix the problems in the FEs, we can mitigate the damage
- Dead time reduction: 6% → 2%



Things that did not work quite as expected – Push based Event Building

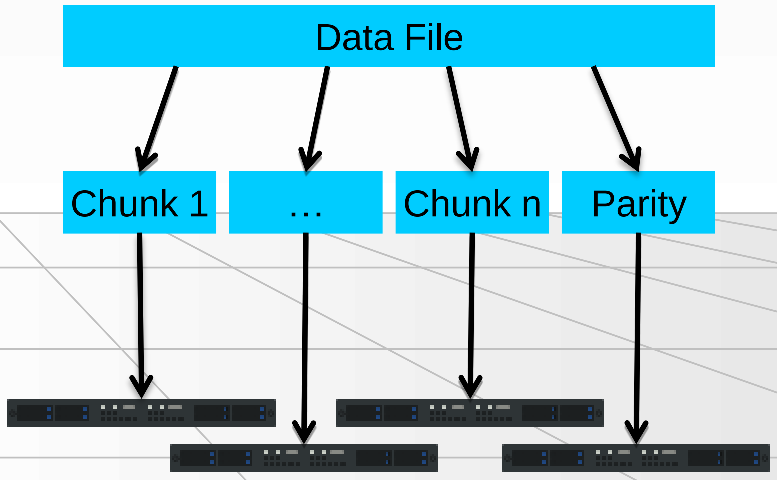


Situation for single event to single farm node

- More of an "expected, but not at this magnitude" thing
- Ingredients:
  - Read-out boards are based on FPGAs
  - Aggregate O(10) event fragments and just drop them onto the wire
  - No resend, no explicit flow control
  - Rely on network hardware with large buffers to handle traffic flow pattern
  - Run at > 90% link load
- Result:
  - Link aggregation load balancing is not 100% fair nor standardized
  - Head of line blocking within sub-farms
  - 1 Gbit/s != 1 Gbit/s → There is a small +- which can cause trouble at very high link loads
  - Output buffers are sufficient on average but are shared between ports
  - **"Time structure analysis of the LHCb DAQ network"** → Today, 15:00 - Poster Session
- Managed to get drop rate down to order of once per hour
- Lot of work and effort
- →Commercial will most of the time not work off the shelf
- After LS2:
  - Computer based ROBs
  - Reliable Event Building network protocol

Things that did not work quite as expected – Deferred Trigger + Unprotected Disks

- Deferred trigger data is transient and stays on disk only for a relatively short time
- Disks are usually not completely full
- We accept, that if a disk breaks, we lose the data on it
- Node level disk redundancy is hard to justify
  - Not enough disk slots for Raid 5+
  - Raid 1 seems a bit of a waste considering the volatility of the data
- Disk hard failure rate is actually very low
- However: Soft failures rate is quite high
  - File System goes to read-only mode
  - In order to not process the same event twice, files are *open()*ed and immediately *unlink()*ed
  - File stays in limbo until the trigger job calls *close()*
  - →Does not work if FS is read-only!
- Cluster File Systems?
  - Either replication (Raid 1) or assume that disk back-end is already protected
  - Too wasteful
- Future: Write our own distributed DAQ file system
  - **"ECFS: A decentralized, distributed and fault-tolerant FUSE file system for the LHCb online farm"** → Unfortunately right now



LS1, Run 2 and beyond

Operational Changes – Virtual Machines

- Modern Servers are too powerful for controls purposes
  - People like to confine different sub-sections of the control system by using different servers
  - A lot of potential CPU power is wasted because the granularity of a single machine is quite large
- Virtual Machines are a way out of this
  - Still strong borders between sub-systems
  - Less physical servers
  - Higher availability through live migration
- Challenges
  - Control system machines eventually have to connect to real hardware
    - Use hardware interfaces that are Ethernet based
    - Limited amount of small machines that act as Hardware ⇔ Ethernet bridges
  - Upside: Good motivation to get rid of most of the Windows machines in our system
  - Less physical servers, but infrastructure becomes more complicated
- **"Performance evaluation and capacity planning for a scalable and highly available virtualization infrastructure for the LHCb experiment",**
  → Today, 15:00 - Poster Session

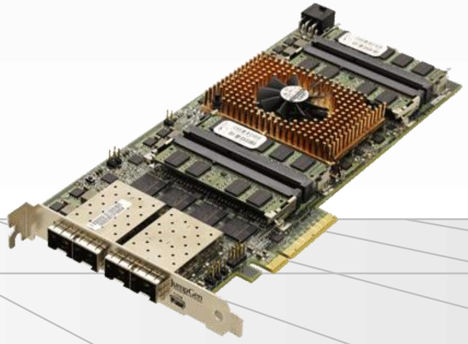
More Operational Changes

- Offline processing on Online Farm during shutdown
  - Online Farm represents a significant computing site
  - Turned farm into a target for our Offline job scheduling system
  - Currently running simulation jobs
- More deferred triggering
  - Currently storing a lot of data that is thrown away by the trigger process later
  - Instead: run a fast selection before storing data and run more detailed processing later/in parallel
  - Allows better usage of the available disk space because data rate has been reduced
  - **“Deferred High Level Trigger in LHCb: A Boost to CPU Resource Utilization”**
    → Tue, 15:45 – This Track

LS2 and beyond

- Trigger free read-out
  - Currently the Online Farm is located close to the detector
  - Need to move upstairs due to power and cooling constraints
  - How do we transport 32 Tbit/s of data over 300 m without going bankrupt?
  - How do we solve the Event Building Problem at these data rates?
  - “**DAQ Architecture for the LHCb Upgrade**”
    → Today, 15:00 – This Track

- Move the Read-out Boards closer into the realm of COTS
  - PCIe based ROB
  - Can be mounted inside a computer
  - The future of networks is hard to predict
  - Gives more options for adopting future network technologies
  - Allows more intelligent Event Building protocols
  - **“A PCIe Gen3 based readout for the LHCb upgrade”**
    → Tue, 14:10 – This Track



- Alternative computing architectures
  - GPUs: “**A GPU offloading mechanism for LHCb** “
    → Today, 15:00 - Poster Session
  - ARM: “**Measurements of the LHCb software stack on the ARM architecture**”
    → Tue, 16:10 – Software Engineering, Parallelism & Multi-Core

Conclusion

- The LHCb Data Acquisition has outperformed its original design specs by more than a factor two, more in certain areas
- Made possible by
  - using our available computing resources to their fullest
  - adopting automation and high availability techniques
  - a lot of hard work by everybody involved
- It was not always smooth sailing
- We learned many lessons from our current system
- We will employ those lessons for future improvements of the DAQ

Thank you for your attention

- Advertisements
  - **"ECFS: A decentralized, distributed and fault-tolerant FUSE file system for the LHCb online farm"**
  - **"DAQ Architecture for the LHCb Upgrade"**
    → Today, 15:00 – This Track
  - **"Time structure analysis of the LHCb Online network"**
    → Today, 15:00 - Poster Session
  - **"Performance evaluation and capacity planning for a scalable and highly available virtualization infrastructure for the LHCb experiment",**
    → Today, 15:00 - Poster Session
  - **"A GPU offloading mechanism for LHCb "**
    → Today, 15:00 - Poster Session
  - **"Phronesis, a diagnosis and recovery tool for system administrators"**
    → Today, 15:00 - Poster Session
  - **"A PCIe GEn3 based readout for the LHCb upgrade"**
    → Tue, 14:10 – This Track
  - **"Deferred High Level Trigger in LHCb: A Boost to CPU Resource Utilization"**
    → Tue, 15:45 – This Track
  - **"Measurements of the LHCb software stack on the ARM architecture"**
    → Tue, 16:10 – Software Engineering, Parallelism & Multi-Core