The LHCb Data Acquisition during LHC Run 1

F Alessio¹, L Brarda¹, E Bonaccorsi¹, D H Campora Perez¹, M Chebbi¹, M Frank¹, C Gaspar¹, L Granado Cardoso¹, C Haen¹, E v. Herwijnen¹, R Jacobsson¹, B Jost¹, N Neufeld¹, R Schwemmer¹, V Kartik¹ and A Zvyagin¹

¹ CERN, CH-1211 Geneva 23, Switzerland
E-mail: rainer.schwemmer@cern.ch

Abstract. The LHCb Data Acquisition system reads data from over 300 read-out boards and distributes them to more than 1500 event-filter servers. It uses a simple push-protocol over Gigabit Ethernet. After filtering, the data is consolidated into files for permanent storage using a SAN-based storage system. Since the beginning of data-taking many lessons have been learned and the reliability and robustness of the system has been greatly improved. We report on these changes and improvements, their motivation and how we intend to develop the system for Run 2. We also will report on how we try to optimise the usage of CPU resources during the running of the LHC (“deferred triggering”) and the implications on the data acquisition.

1. Introduction to the LHCb DAQ

The design of LHCb DAQ is based on using industry standard components were ever possible, minimising the number of components and keeping the protocols as simple as possible [1].

1.1. Original architecture

The original LHCb data acquisition system was designed for 1 MHz of events of 35 kB and an event-filter farm of about 1000 server-nodes for event-filtering, distributed in 50 sub-farms. For cost-saving reasons the original system was using a single core router, to which the 300 read-out boards were directly connected. The “other” side of the router was connected to the 50 distribution switches to which the server nodes of the 50 sub-farms were connected. An illustration of the system can be seen in figure 1a. While conceptually scalable this configuration has some practical limitations: the approximately 600 ports in the core router available for read-out boards and the number of ports in the aggregation layer for the filter-nodes.

The protocol used in LHCb is a simple push-protocol with a central, credit-based load-balancing [2]. For this load-balancing the Timing and Fast Control system is used, which has a direct, reliable, out-of-band connection to the read-out boards. This connection is used to transmit the IP address of the destination event-filter node to the read-out boards [3]. Since this information arrives synchronously the data packets from the read-out boards hit the core router synchronously. This leads to a significant instantaneous over-commitment at the output of the router, which can only be handled with large buffers. For simplicity no re-sends are implemented, any loss of any packet from any source leads to total loss of the entire events, as incomplete events are not accepted by the LHCb trigger software. Minimising packet-loss
is therefore key for the success of LHCb’s data acquisition, probably even a bit more than for other LAN-based DAQ systems.

For storage we chose a high-end storage device from the HPC world, which provides excellent streaming bandwidth required for our file-writing at reasonable cost and with the additional advantage that its performance does not suffer under disk-rebuilds [4].

Figure 1: Comparison of original and current DAQ network architecture. Since the original design we have added a second core router to accommodate more read-out boards, hard disks to most of the filter servers and a high availability SAN as storage back-end.

1.2. Evolution of requirements
During the LHC Run 1 all key parameters increased significantly. Running at higher luminosity than originally foreseen and at the same time with a 50 ns bunch-spacing rather than 25 ns, increased the event-size to approximately 60 kB. Several high-occupancy detectors needed more read-out boards to cope with the higher data-rates. Also the single router, while normally quite reliable remained a single point of total failure. Changes in the physics programme led to a significant increase in data output to permanent storage from about 1 kHz to about 5 kHz and 75 MB/s to almost 300 MB/s.

1.3. Current architecture
We exploited the fact that the read-out-boards have 4 independent links to connect to the network, and that full connectivity in the network is only required for the read-out-boards, but not for the filter-farm nodes. A second identical core router was acquired and each read-out-board connects now with either one or two links to each core router. Each core router connects to each distribution-switch, with half the available links. This increases the total available bandwidth and buffer-space in the network significantly and has proven to be sufficient for the needs of LHCb.
New sub-farms were added for a total of 56, because the total bandwidth available into the existing sub-farms was limited by the physical connectivity (the number of cables installed between the core and the distribution switches) and this could not be easily increased. The new farms are connected to the cores with 10 Gigabit Ethernet, which by 2011 had become affordable.

More farm-nodes have been almost constantly added as funds became available. This could be done even during physics without any interference to the data-taking. Since the nodes are disk-less and boot from a network server, they simply need to be declared to the control-system and booted. They then send credit-requests and can immediately start processing data.

The storage back-end has been upgraded to one with higher performance to cope with increased demand due to the increased output rate.

Dataflow inside a filter node during beam time. Dataflow inside a filter node while accelerator is off.

Figure 2: Dataflow inside the filter nodes during and after LHC beam time. While the accelerator is running, a fraction of the produced data is stored on local disks on the filter node. When the accelerator is off, the data is recalled from disk and processed.

1.4. System optimisation

An important change was introduced in the beginning of 2012 [6], illustrated in figure 2. To leverage the low duty cycle of the LHC, LHCb has started to store events, which cannot be processed online by the high-level trigger on local disks of the event-filter servers. This events are then processed after the LHC has stopped delivering live collisions. More than 30% CPU-power could be gained in this way. The existing disks were very small, because they were only used as a swap-space. A massive upgrade was performed where more than 1000 1 TB disks were added to the system. The total capacity of this distributed storage allows us to buffer up to 5 hours of untriggered detector data.

Another field of optimisation is the LHCb Experiment Control System (ECS). The ECS has proven to be very flexible and adaptable. All the changes described so far could be accommodated, without changing any fundamental aspect. With increased experience with the detector and the LHC operations, the automation of the operation of the experiment has been pushed forward [7]. The enormous effort put into automation payed of as very high operational efficiency of the experiment. As shown in figure 3, operational dead-time was reduced down to 2%. 
Figure 3: Graphical Front-end for the automation layer of the control system. The accelerator state dictates the states of the detector. The operator is only asked for confirmation on sensitive state changes and critical errors that can not be recovered automatically.

2. What went wrong
In this section we are going to take a look at the major things that did not quite go according to plan and present how we overcame these problems.

2.1. Dead-time free read-out
A major design goal of the LHCb experiment was to provide a quasi dead-time free read-out at a trigger rate of 1 MHz. Since a typical event read-out cycle takes 40 LHC clock cycles, data for consecutive events needs to be temporarily stored on the front-end. To achieve this, the front-ends are all equipped with deep derandomizing buffers, which can store up to 16 consecutive events. Together with the predicted rate for interesting events and the trigger efficiency and rate this leads to a dead-time of approximately 1%. To prevent a possible 17th trigger from reaching the front-end, the read-out supervisor is equipped with a simple, token like emulator to determine the filling level of the front-end. For every trigger it puts 40 tokens on a counter, which is decreased by 1 for every clock cycle. If the counter would go over 40 x 16, the following trigger signals would be suppressed until there is sufficient space again.

While the front-ends all fulfil the memory requirements for the case of 16 consecutive triggers, they sometimes run short on memory cells due to problems with the memory allocation algorithm. Cells that are available for buffer space can not be allocated because they are mistakenly still flagged as used. The result is usually a hard crash of the front-end chips which needs a power-up reset in order to restore the chips to working order. On top of this, there are several chip types for the different sub-detectors, which behave differently and are sensitive to different trigger sequences.

Figure 4 shows the current and past state of the system. On the x-axis we show the trigger rate and on the y-axis we show the dead-time. The red curve represents the dead-time that was found to be safe with using the simple token based emulator. The green curve shows the design goal. To mitigate the discrepancy, we integrated the relevant parts of the front-end ASIC designs as emulators into the firmware of the read-out supervisor. The result is the curve in pink which improves the dead-time for our current working point at a luminosity of \(4.0 \times 10^{32} \text{cm}^{-2} \text{s}^{-1}\) from 6% to 2%.
2.2. Push based event building

As shown in figure 1a, the LHCb DAQ network was originally designed as a fully cross connected 1 Gbit/s Ethernet network. Data flows from the read-out boards to the event builders as raw IP packets with a minimal header which allows the event builder to know where a certain fragment comes from and to which event it belongs. Once the packet has been sent, the read-out board will immediately remove the packet from its output buffer and forget about it. There is no mechanism foreseen to repeat packets that might have been lost somewhere on the way between source and receiver. To keep the overall costs of the network low, the source output links are used to their limit and are in some high occupancy cases running at more than 90% of the maximum link load.

The read-out boards contain almost no buffers and have to send out their data as soon as the conversion into a network packet has finished. This means they are all sending within a very small time window. Figure 5 illustrates this problem. While the output link to a singular farm node is on average not heavily loaded, there is a temporary over commit when it is the turn of the node to receive new data. The two switches on the way to the node are suddenly faced with 350 packets that all want to go to the same output port at the same time. This requires a lot of buffering in the switches and a lot of research was invested into network equipment during the R&D phase of the experiment to find hardware that actually has the necessary amount of buffer memory [5].

Despite the fact that the selected core switches do have the necessary memory to buffer all the traffic, there were many subtle problems to be fixed to actually be able to use these buffers and reduce packet drop rates. A few select problems we encountered are treated here:

- **Link Aggregation**: As seen in figure 5, there are six physical links which are aggregated into a single virtual link. These links connect the sub-farm fan-out switches with the core network. Link aggregation is not a standard and so there can be problems if equipment of different vendors are used. Another pitfall for link aggregation is the load balancing algorithm. These algorithms are usually not 100% fair and sometimes select the physical link of a packet based on the data inside the packet. To make full use of the link aggregation, we worked together with the switch manufacturer to produce a special switch firmware. It
allows to steer packets to certain physical links depending on information we put into the packets.

- **Head of line blocking:** If two successive events are sent to the same sub-farm, head of line blocking can occur on the aggregated links between core router and fan-out switches. This will usually overflow the output buffers and cause packet drops. We had to change the destination assignment algorithm in our read-out supervisor to accommodate this fact and spread out successive events to different sub-farms.

- **Sharing of buffer memory:** Even though successive events are going to different sub-farms there can still be depletion of output buffer memory, because the memory is often shared between output ports. More adjustments where done to the destination assignment algorithm to circumvent this feature.

- **Oscillator variances:** Switches and network cards are electronic devices and as such they are affected by effects of the underlying electronics. In particular the data transmission speed of a device is determined by a crystal oscillator built into the device. While very precise, these oscillators have of course variances and so a device with 1 Gbit/s is actually running at 1 Gbit/s ± ϵ. The transmission speed in Ethernet is always determined by the sender and consequently there is no common clock that is valid within the entire network. Links between network nodes are all running at different speeds and so the fan-out switches can run into trouble when the core router is sending at a slightly higher speed than the fan-out switches are sending to the filter nodes. If the link load is close to 100%, data will start to accumulate in the fan-out switch and lead to occasional packet losses.

In the end we managed to get the packet drops down to the order of once per hour but the effort we had to put in might be considered disproportional. The lesson learned here is that commercial equipment will most of the time not work off the shelf. In particular not when used at its limits.

We are currently in the R&D phase of the next generation of read-out network for LHCb. In the new system, we will not rely on buffers in network devices and also use a more sophisticated read-out protocol, which can detect occasional packet drops and can re-send data if necessary.

![Figure 5: Push based event building as seen from a single farm node.](image)
3. Conclusion
In this paper we have shown how the LHCb Data Acquisition System evolved from its original concepts to the state it is currently in. The DAQ has outperformed its original design specs by more than a factor of two in some areas and a lot more in others. By developing clever operational procedures like the deferred triggering and relying heavily on automation, we have managed to use our computing resources to their full extent and the delivered beam time to its maximum potential.

Many more minor changes have been done, which can not all be mentioned here, and it was certainly not always smooth sailing. As a result we have come out of Run 1 with a much more robust and efficient system than we started with. We are looking forward to Run 2 of the LHC and are going to continue to employ the lessons learned in the immediate future and beyond.

References
[1] N. Neufeld. The LHC Data Acquisition System: Status and Prospects. In Proceedings of CHEP2009, 2009. invited plenary talk.
[2] F. Alessio et al. LHCb online event processing and filtering. J. Phys. Conf. Ser., 119:022003, 2008.
[3] G. Haefeli et al. The LHCb DAQ interface board TELL1. Nucl. Instrum. Meth., A560:494–502, 2006.
[4] S. S. Cherukuvada and N. Neufeld. High-Performance Storage for LHCb Data Acquisition. IEEE Trans. Nucl. Sci., 55:278–283, 2008.
[5] A. Barczyk et al. High Rate Packets Transmission on Ethernet LAN Using Commodity Hardware. IEEE Trans. Nucl. Sci., 53:810–816, 2006.
[6] Markus Frank, Clara Gaspar, Niko Neufeld, and Beat Jost. Deferred High Level Trigger in LHCb: A Boost to CPU Resource Utilization. In D. Groep, editor, Proceedings of the 20th International Conference on Computing in High Energy and Nuclear Physics (CHEP2013), 2014.
[7] C. Gaspar, F. Alessio, L. Cardoso, M. Frank, J. C. Garnier, E. v. Herwijnen, R. Jacobsson, B. Jost, N. Neufeld, R. Schwemmer, O. Callot, B. Franek. The LHCb Experiment ControlL System: on the path to full automation. In ICALEPCS2011 Contributions to the Proceedings, 2011. available: [Online] http://icalepcs2011.esrf.eu/proceedings.htm.