# CONTROLLING AND MONITORING THE DATA FLOW OF THE LHCb READ-OUT AND DAQ NETWORK

Rainer Schwemmer, C. Gaspar, N. Neufeld, D. Svantesson, CERN, Geneva, Switzerland

## *Abstract*

The LHCb read-out uses a set of 320 FPGA based boards as interface between the on-detector hardware and the GBE DAQ network. The boards are the logical Level 1 (L1) read-out electronics and aggregate the experiment's raw data into event fragments that are sent to the DAQ network. To control the many parameters of the read-out boards, an embedded PC is included on each board, connecting to the boards ICs and FPGAs. The data from the L1 boards is sent through an aggregation network into the High Level Trigger farm. The farm comprises approximately 1500 PCs which at first assemble the fragments from the L1 boards and then do a partial reconstruction and selection of the events. In total there are approximately 3500 network connections. Data is pushed through the network and there is no mechanism for resending packets. Loss of data on a small scale is acceptable but care has to be taken to avoid data loss if possible. To monitor and debug losses, different probes are inserted throughout the entire read-out chain to count fragments, packets and their rates at different positions. To keep uniformity throughout the experiment, all control software was developed using the common SCADA software, PVSS, with the JCOP framework as base. The presentation will focus on the low level controls interface developed for the L1 boards and the networking probes, as well as the integration of the high level user interfaces into PVSS.

## LHCB READ-OUT CHAIN

The LHCb DAQ system is built around an Ethernet based network, which transports the experiment's data from the detector to a Linux based filter farm. This farm consists of 1500 computers and processes the data of every accepted event and then decides whether the event is interesting or if it should be discarded. Accepted events are then sent on via IP over Ethernet to a group of computers which replicates the stream of accepted data to the disk writing processes and to a dedicated monitoring farm.

Data is accepted at a maximum rate of 1.1 MHz which translates to a data rate of approximately 50 GByte/s on the input of the network. On the border between network and detector is a set of 350 FPGA based converter boards – Trigger Electronics and Level 1 board (TELL1) [1] – which receive the data via optical links from the detector. The data is first checked for consistency and then compressed into a sub-detector specific format. After that, the event data is combined with meta information and a unique event ID. In the last step, data is coalesced into IP packets and sent to an on-board, 4 port Gigabit Ethernet (GbE) card. The TELL1s then send the data to the filter farm through two main data switches and a set of 50 fan-out switches. The total packet rate in the network is about 35 MHz.

Inside a filter farm node, the event fragments are assembled into complete events and then handed off to trigger applications which do a partial reconstruction and filter process. From the filter farm onward, the data rate drops to a more manageable rate of 200 MB/s and is again aggregated on two dedicated machines before being replicated to the online monitoring farm and the storage layer. The local storage system is only a temporary buffer for the data though. Data is written in sets of 3 GB files. Once a file is full, its consistency is verified, check sums are calculated and the file is sent off to permanent tape storage at CERN.

Throughout this chain, there are many points, at which data can be sporadically lost or which can halt the data taking process completely.

## TYPICAL LOSS POINTS AND BOTTLENECKS

### *Read-out Boards*

When a read-out board fails to produce data, there will not be enough fragments to rebuild the event in the farm and the event will be discarded. If a TELL1 stops sending any data at all, the whole read-out chain is stopped. If it cannot cope with the data rate for some reason, and slows down, the whole DAQ process is affected. In order to quickly diagnose problems inside the read-out board, the data flow is already monitored on its way through the board.

There are five FPGAs on each board, where four are responsible for checking and compressing the sub-detector data and one is assembling the data from the four preprocessors. Additional meta information from the Timing and Fast Control (TFC) system is then added to each event, before it is sent to a buffer for network transmission.

Typical problems here can be a front-end not sending data or the data transmission over the board between chips. The compression and gathering algorithms, if they are overwhelmed by too much data due to noisy channels in the detector or unexpectedly high occupancy events can also slow down the process or fail completely.

### *Network*

At the design time of the read-out system it was deemed too unpractical to have a reliable transport protocol between the TELL1 boards and the filter farm. The huge number of farm nodes and TELL1 boards would necessitate a substantial amount of memory for state information and data buffering on the read-out boards, which at that time was just not an option. The chosen protocol is a pure push protocol, where data is being sent as IP packets whenever bandwidth is available on the read-out boards. In a way the protocol behaves like UDP. In case the switches decide to drop packets, there is no retransmission.

The situation is further complicated by the fact, that all the fragments for a particular event have to be sent to a single farm node. The fragments leave the TELL1 boards all at the same time and for a short time there is a strong overcommitment of the output port to the farm node where the data is supposed to go to. The average rate to a single farm node is still far below the 1 Gigabit/s limit, but for a switch this is a typical situation to throw away packets, if it runs out of buffer space.

### *Farm Nodes*

Inside the farm computers, the data is received by the network card, assembled into events and then passed on to the trigger applications[4]. If the machine is too busy, data can be lost already on it's way from the network card to the kernel, before it enters any application. Other problems can be trigger processes getting stuck and not processing events any more, or a machine in general having hardware problems which influences performance.

### *Storage Layer*

As the data rate is reduced to about 200 MB/s after the trigger farm and there is only inter-computer communication, the read-out chain uses the TCP/IP protocol from the farm onward. TCP is a reliable protocol, so data loss in the network is not as much of a problem any more as in the steps before. However, switches can fail and processes can die unexpectedly, so this part of the system is monitored as well.

The actual process of writing at a sustained rate of 200 MB/s for several hours per day also requires special hardware, especially if it has to be protected from single mode failures. For data storage, we use a fiber channel based Storage Area Network (SAN) consisting of several head nodes and a fully redundant hard-disk array. The entire SAN is set up in an active-active configuration and again each connection point in the fiber network is monitored to make sure that all components continue working and share the load.

## READ-OUT CHAIN PROBES

In the end one wants to be able to diagnose, as quickly as possible, why the system's performance is degraded and which link in the read-out chain is responsible. In order to find the culprit, we have – over time – inserted many probes to gather statistics throughout the entire DAQ process. At the same time, care had to be taken to not overload the components with probes, because they are usually all highly utilized by the DAQ process.

### *Read-out Boards*

On the TELL1 boards, the number of event fragments that come in on every optical link is already counted, to find misbehaving front-ends. After the pre-processing stage, event numbers are counted as they leave the pre-processing FPGAs and when they enter the aggregation FPGA. A final count is done when they are assembled into IP packets and how many packets are sent to the GbE card.

The TELL1 comes with an embedded 486 CPU that sits on each board and which is capable of talking to all the FPGAs and other micro-chips on the board via PCI and $I^2C$ buses. A program on the PC periodically gathers all the counters and publishes them through a unified messaging layer, called DIM[2], to the Experiment Control System (ECS) [3].

### *Network*

Since we are using only off-the-shelf network equipment we can use the standard counters that come with the switches and routers. What makes the situation a bit more complicated is the fact that a switch does not know anything about event fragments but only about network packets. It also has persistent counters, that are not reset on every run change. Additionally, on the core routers, which have to cope with the full 50 GB/s, 35 MHz packet rate, we cannot permanently run the full statistics gathering, because it overloads their CPUs and slows down normal network operations.

Nevertheless, this is the only semi-blind spot in the system, and by monitoring the data leaving the TELL1s as network packets and the data entering the farm nodes also as packets, we can trace back problems to the switches and then check in more detail where the problem is by switching on specific counting features in the switches.

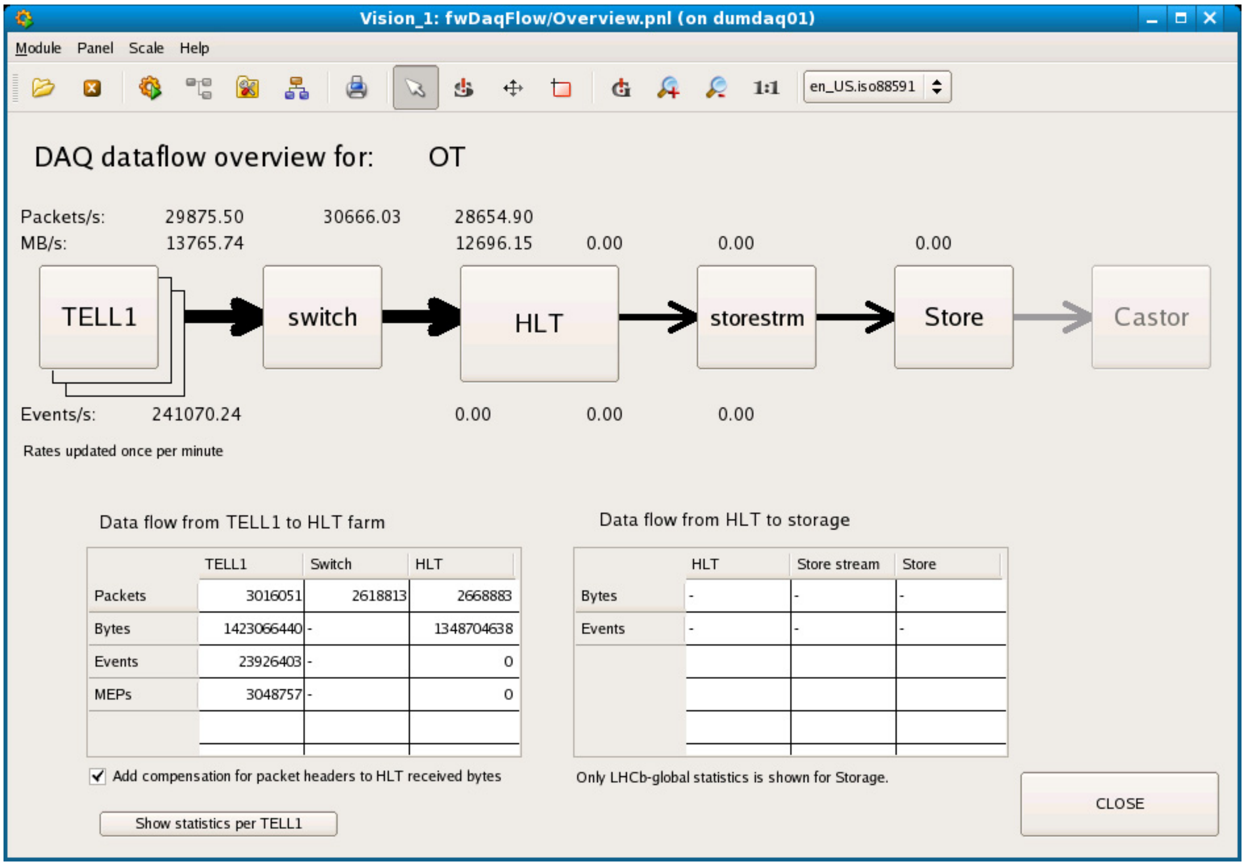
General data, like port counters are directly gathered via scripts from the ECS. For more in depth debugging sessions, the system administrators have to directly log in to the switches and check the statistics manually or use dedicated console scripts.

### *Farm Nodes*

We also had trouble with data being lost on the way from the network card to the application. We are using the counting features of the Linux Firewall to count packets as they are arriving. The packets are filtered by source IP addresses, which can then be mapped to the sending TELL1s.

Further up in the chain we count again the number of assembled event fragments and from which source they came. Finally, the total processed and accepted/rejected events are counted as well.

Again, like on the read-out boards, a special program was developed to gather the statistical counters of the fire-



DAQ dataflow overview for: OT

| | TELL1 | switch | HLT | storestrm | Store | Castor |
|---|---|---|---|---|---|---|
| Packets/s: | 29875.50 | 30666.03 | 28654.90 | | | |
| MB/s: | 13765.74 | | 12696.15 | 0.00 | 0.00 | 0.00 |
| Events/s: | 241070.24 | | 0.00 | 0.00 | 0.00 | |

Rates updated once per minute

Data flow from TELL1 to HLT farm

| | TELL1 | Switch | HLT |
|---|---|---|---|
| Packets | 3016051 | 2618813 | 2668883 |
| Bytes | 1423066440 | - | 1348704638 |
| Events | 23926403 | - | 0 |
| MEPs | 3048757 | - | 0 |

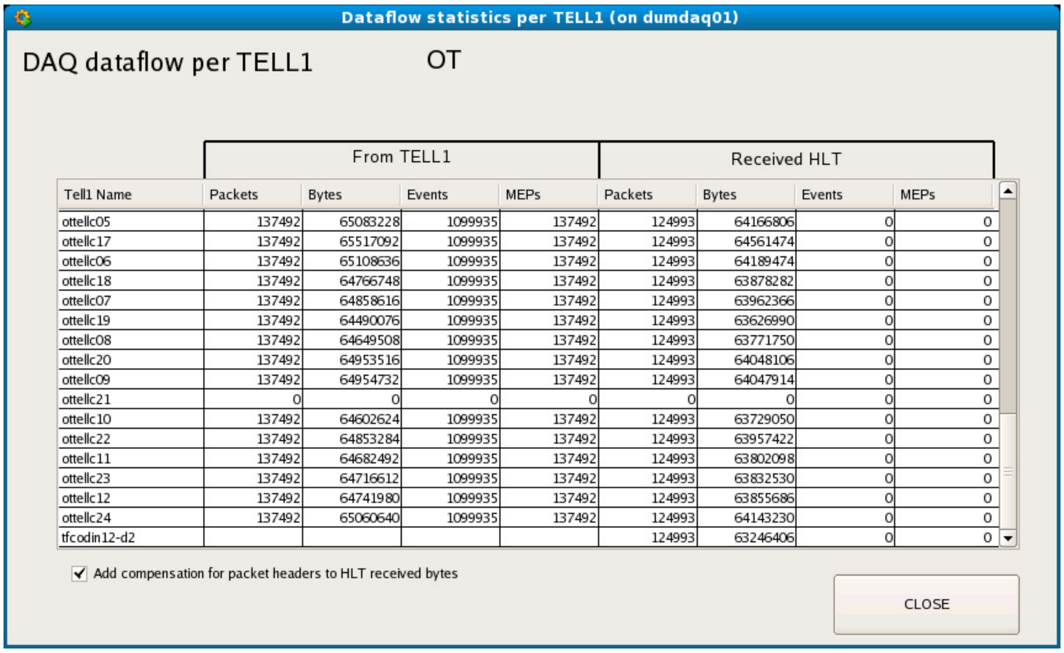
[✓] Add compensation for packet headers to HLT received bytes

Data flow from HLT to storage

| | HLT | Store stream | Store |
|---|---|---|---|
| Bytes | - | - | - |
| Events | - | - | - |

Only LHCb-global statistics is shown for Storage.

Figure 1: Top overview panel of the DAQ flow monitor. The User can click on the buttons of the different routing points to get a more a more detailed view of the subsystem.



DAQ dataflow per TELL1 OT

| | From TELL1 | | | | Received HLT | | | |
|---|---|---|---|---|---|---|---|---|
| Tell1 Name | Packets | Bytes | Events | MEPs | Packets | Bytes | Events | MEPs |
| ottellc05 | 137492 | 65083228 | 1099935 | 137492 | 124993 | 64166806 | 0 | 0 |
| ottellc17 | 137492 | 65517092 | 1099935 | 137492 | 124993 | 64561474 | 0 | 0 |
| ottellc06 | 137492 | 65108636 | 1099935 | 137492 | 124993 | 64189474 | 0 | 0 |
| ottellc18 | 137492 | 64766748 | 1099935 | 137492 | 124993 | 63878282 | 0 | 0 |
| ottellc07 | 137492 | 64858616 | 1099935 | 137492 | 124993 | 63962366 | 0 | 0 |
| ottellc19 | 137492 | 64490076 | 1099935 | 137492 | 124993 | 63626990 | 0 | 0 |
| ottellc08 | 137492 | 64649508 | 1099935 | 137492 | 124993 | 63771750 | 0 | 0 |
| ottellc20 | 137492 | 64953516 | 1099935 | 137492 | 124993 | 64048106 | 0 | 0 |
| ottellc09 | 137492 | 64954732 | 1099935 | 137492 | 124993 | 64047914 | 0 | 0 |
| ottellc21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| ottellc10 | 137492 | 64602624 | 1099935 | 137492 | 124993 | 63729050 | 0 | 0 |
| ottellc22 | 137492 | 64853284 | 1099935 | 137492 | 124993 | 63957422 | 0 | 0 |
| ottellc11 | 137492 | 64682492 | 1099935 | 137492 | 124993 | 63802098 | 0 | 0 |
| ottellc23 | 137492 | 64716612 | 1099935 | 137492 | 124993 | 63832530 | 0 | 0 |
| ottellc12 | 137492 | 64741980 | 1099935 | 137492 | 124993 | 63855686 | 0 | 0 |
| ottellc24 | 137492 | 65060640 | 1099935 | 137492 | 124993 | 64143230 | 0 | 0 |
| tfcodin12-d2 | | | | | 124993 | 63246406 | 0 | 0 |

[✓] Add compensation for packet headers to HLT received bytes

Figure 2: Detailed counters of the intersection between TELL1s and the trigger farm. ottellc21 is not producing any data and subsequently no events are produced.

wall and the trigger processes and publish them to the ECS. In this case more effort had to be put into the program, since it has to process several times the number of fragments times the number of TELL1s times the number of farm nodes.

We developed a tiered process hierarchy, where programs on individual farm nodes send their statistical information to higher level nodes where the data is added together and then sent to a central processing program, which again adds the sub-results and publishes them. The gathering programs also tap into the control system to detect when runs start and stop, and to detect which parts of the detector are actually part of the read-out. This is necessary, because the detector can be split into its individual sub-detectors which can run in standalone mode.

### *Storage Layer*

Finally the accepted data has to be written to disk. Again there are a number of critical processes, which can fail and contribute to the loss of data. Since this part of the read-out is based on TCP, data loss is not caused by network packet drops or disappearing events, but through loss of performance. If the data cannot be written to the disks fast enough, or there are general hardware problems on the way to the disks, the system will slow down, decreasing the amount of physics data that is gathered over time.

Data arrives from the farm on two dedicated machines, which aggregates the events from the 1500 sources into file streams. There are usually several files being written in parallel for better load balancing. The files are then written by a simplified distributed file system [5]. All written events are again counted by the writing daemons and published to the ECS.

While we are still counting events up to the point where the data is written into actual files, we switch to purely rate based monitoring on the actual SAN back-end. There are fairly good standard tools like Munin[6] and problems in this area are outside the scope of what the operators of the experiment are usually dealing with. Subsequently we are using these tools directly instead of relaying the information to the ECS. Information is gathered from the storage head nodes directly, on the Fiber Channel switches and from the disk storage system itself. We developed dedicated plug-ins for munin, to monitor individual raid sets in the SAN and the throughput through each individual fiber channel path that leads to them. The results are then displayed on a dedicated web page. A permanent screen in the control room displays a sub-set of the most important rates to the operators.

## USER INTERFACE

All the information gathered from the different counters and probes throughout the network has to be compiled into an easily understandable interface, which allows the user to quickly pinpoint problems. To hide the fact that there are hundreds of thousands of counters, we developed a dedicated master process, which gathers all the counters, sorts them into categories of either event counters, network packet counters or byte counters and calculates the rates of these for all the major routing points in the system.

A UI panel then shows a schematic representation of these points and the rates at which data is passing through them. In case of problems, the operator can quickly diagnose the point at which the disruption is happening. For more in-depth analysis, all the routing points have dedicated UI expert panels, which show the full multitude of counters. The top overview panel is shown in Fig. 1. In the case shown here, there is a problem with the assembly of the event fragments. Packets are arriving at the farm, but no events are produced by the event building processes. Looking at the detailed view of the TELL1 - HLT connection in Fig. 2, one can see a single TELL1 not producing any data.

## CONCLUSION

We have developed a multi-tiered hierarchy of programs, which are capable of reading the various statistics counters that are present throughout the LHCb DAQ chain and moreover, are capable of coping with the enormous amount of information. The data is brought into a unified format which makes it easier to spot points of data loss in the system.

All the counters are analyzed by a dedicated process and then presented to the user in a simple panel, which hides most of the complexity of the system, but gives access to more detailed information if necessary. This system has helped us to quickly find problematic network links, failing read-out boards and front-ends and has improved the response time from the time a problem occurs to being fixed.

## REFERENCES

[1] Guido Haefeli et al., “TELL1 Specification for a common read out board for LHCb”, IPHE Note 2003-02, LHCb Note 2003-007.

[2] C. Gaspar et al., “DIM, a Portable, Light Weight Package for Information Publishing, Data Transfer and Inter-process Communication”, Proc. of CHEP 2000, Padova.

[3] C. Gaspar, B. Franek, R. Jacobsson, S. Morlini, N. Neufeld, and P. Vannerem, “An integrated experiment control system, architecture, and benefits: The LHCb approach”, IEEE Trans. Nucl. Sci., vol. 51, no. 1, pp. 513520, Jun. 2004.

[4] G. Barrand et al., “GAUDI - A software architecture and framework for building LHCb data processing applications”, Proc. of CHEP 2000, Padova.

[5] J. Garnier, N. Neufeld, and S.S. Cherukuwada, “Non-POSIX File System for LHCb Online Event Handling”, Real Time Conference (RT), 2010 17th IEEE-NPSS, p 1-4.

[6] Munin: “A networked resource monitoring tool that can help analyze resource trends”, http://munin-monitoring.org/