

The CMS event builder and storage system

G Bauer⁷, B Beccati³, U Behrens², K Biery⁶, A Brett⁶, J Branson⁵,
E Cano³, H Cheung⁶, M Ciganek³, S Cittolin³, J A Coarasa^{3,5},
C Deldicque³, E Dusinberre⁵, S Erhan⁴, F F Rodrigues¹, D Gigi³,
F Glege³, R Gomez-Reino³, J Gutleber³, D Hatton², M Klute⁷,
J-F Laurens³, C Loizides⁷, J A Lopez Perez^{3,6}, F Meijers³, E Meschi³,
A Meyer^{2,3}, R K Mommsen⁶, R Moser^{3a}, V O'Dell⁶, A Oh^{3b},
L Orsini³, V Patras³, C Paus⁷, A Petrucci⁵, M Pieri⁵, A Racz³,
H Sakulin³, M Sani⁵, P Schieferdecker^{3c}, C Schwick³,
J F Serrano Margaleff⁷, D Shpakov⁶, S Simon⁵, K Sumorok⁷ and
M Zanetti³

¹ Centro Federal de Educação Tecnològica Celso Suckow da Fonseca, Rio de Janeiro, Brazil

² DESY, Hamburg, Germany

³ CERN, Geneva, Switzerland

⁴ University of California, Los Angeles, California, USA

⁵ University of California, San Diego, California, USA

⁶ FNAL, Chicago, Illinois, USA

⁷ Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

^a Also at University of Technical University of Vienna, Vienna, Austria

^b Now at University of Manchester, Manchester, United Kingdom

^c Now at University of Karlsruhe, Karlsruhe, Germany

E-mail: remigius.mommsen@cern.ch

Abstract. The CMS event builder assembles events accepted by the first level trigger and makes them available to the high-level trigger. The event builder needs to handle a maximum input rate of 100 kHz and an aggregated throughput of 100 GB/s originating from approximately 500 sources. This paper presents the chosen hardware and software architecture. The system consists of 2 stages: an initial pre-assembly reducing the number of fragments by one order of magnitude and a final assembly by several independent readout builder (RU-builder) slices. The RU-builder is based on 3 separate services: the buffering of event fragments during the assembly, the event assembly, and the data flow manager. A further component is responsible for handling events accepted by the high-level trigger: the storage manager (SM) temporarily stores the events on disk at a peak rate of 2 GB/s until they are permanently archived offline. In addition, events and data-quality histograms are served by the SM to online monitoring clients. We discuss the operational experience from the first months of reading out cosmic ray data with the complete CMS detector.

1. Introduction

The CMS event builder collects event fragments from approximately 500 front-end readout modules (FEDs) located close to the underground detector and assembles them into complete

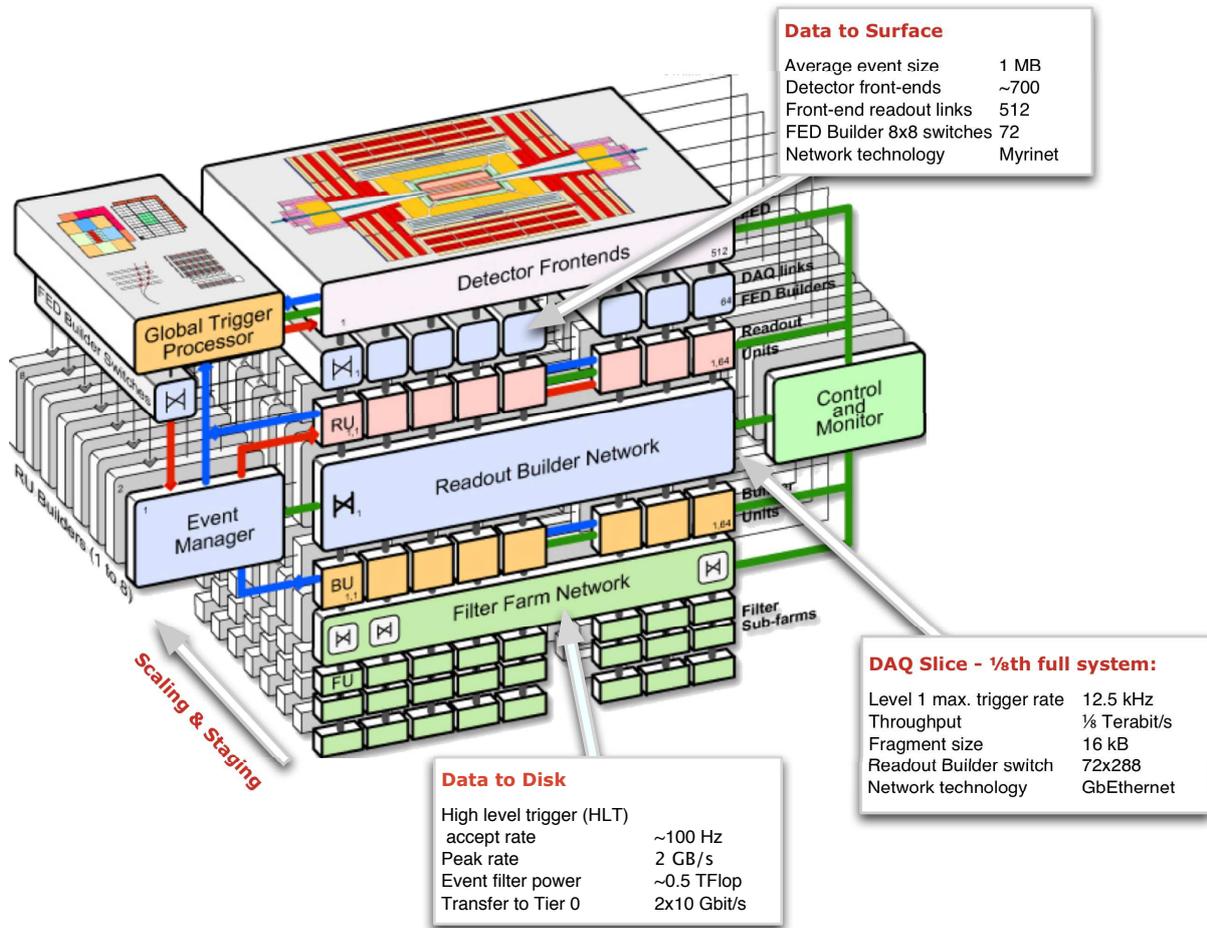


Figure 1. Overview of the CMS DAQ system consisting of detector front-end readout modules (FEDs) feeding up to 8 DAQ slices consisting of independent event builders and high-level trigger (HLT) farms.

events at a maximum rate of 100 kHz (see figure 1). The complete events are handed to the high-level trigger (HLT) processors located on the surface and running offline-style algorithms to select interesting events. Accepted events are transferred to the storage manager (SM) which temporarily stores the events on disk at a peak rate of 2 GB/s until they are permanently archived offline. In addition, events and data-quality histograms are served by the storage manager application to online monitoring clients.

The design of the CMS Data Acquisition System and of the High Level Trigger is described in detail in the Technical Design Report [1].

2. CMS Event Building – 2 Stages

CMS employs only 2 trigger levels: a first level trigger based on custom hardware, and a high-level trigger (HLT) using commodity PCs.

The assembly of event fragments into complete events takes place at a level-1 accept rate of 100 kHz. In order to cope with the aggregated bandwidth of 100 GB/s, and to provide a scalable system, a 2-tier approach for the event building was chosen: an initial pre-assembly by the FED builder and a final assembly and event selection by several independent DAQ slices.

2.1. FED Builder

Each FED builder assembles data from 8 front-end links into one super-fragment using redundant Myrinet [2] switches. The super-fragment is delivered to one of up to 8 independent readout slices on the surface, where it is buffered on commodity PCs, the readout units (RUs). For a detailed discussion of the FED builder, see [3].

2.2. DAQ Slices

The DAQ slices are mutually independent systems, which are in charge of building full events out of super-fragments, performing the physics selections, and forwarding the selected events to mass storage (SM). The independence of each DAQ slice has several advantages:

- Relaxed requirements: Each slice needs to handle a reduced event rate of 12.5 kHz instead of 100 kHz
- Scalability: Slices can be added or removed as needed
- Robustness: In case that one slice fails, data taking does not stall as other slices continue to process events
- Reduced interconnects: Reduces the number of interconnected components and less internal couplings.
- Technology independence: Different technologies can be used for the FED builder and for the DAQ slice, or the technology can even differ between DAQ slices

Each readout unit (RU) is connected with two or three links (“rails”) via a large switch to builder units (BUs). The existing system employs the TCP/IP protocol over Gigabit Ethernet [4]. In order to achieve the necessary performance, a specific configuration of the source and destination VLANs in the network is required. The event building in each slice is controlled by one event manager (EVM), which receives the level-1 trigger information. The RUs, BUs, and the EVM make up the RU-builder. The BUs store the complete events until the filter units (FUs) running the high-level trigger (HLT) algorithms either rejected or accepted the events. Accepted events are sent to the storage manager (SM), which writes the events to disk.

2.3. RU-Builder Protocol – Token Passing

The RU-builder consists of 3 separate services (see figure 2): the readout unit (RU) buffers the event fragments for the assembly, the builder unit (BU) assembles the event, and the event manager (EVM) interfaces with the trigger and orchestrates the data flow. The applications exchange messages over the network and use First-In-First-Out queues (FIFOs) to keep track of requests, trigger data, and event data.

The BU can build several events concurrently (see figure 3). An event is composed of one super-fragment from the trigger and one super-fragment from each RU that takes part in the event building. If the BU has resources available to build another N events, it sends the EVM N tokens representing freed event IDs (1). The EVM sends the trigger supervisor N new trigger credits. If the trigger has credits available, it will send the trigger information of new events to the EVM. The EVM pairs the trigger information with an available token and notifies the RUs to readout the event’s data (2). Several tokens together with the corresponding trigger data are sent to the BU requesting new events (3). The BU adds the trigger data as the first super-fragment of the event, and requests the RUs to send the rest of the event’s super-fragments (4). Each RU responds by retrieving the super-fragment from its internal fragment look-up table and sends it to the BU (5). The BU builds the super-fragments that it receives from the RUs into a whole event. Filter units (FUs) can ask the BU to allocate them events. If the BU has a complete event, it assigns a FU a complete event (6). When a FU has finished processing

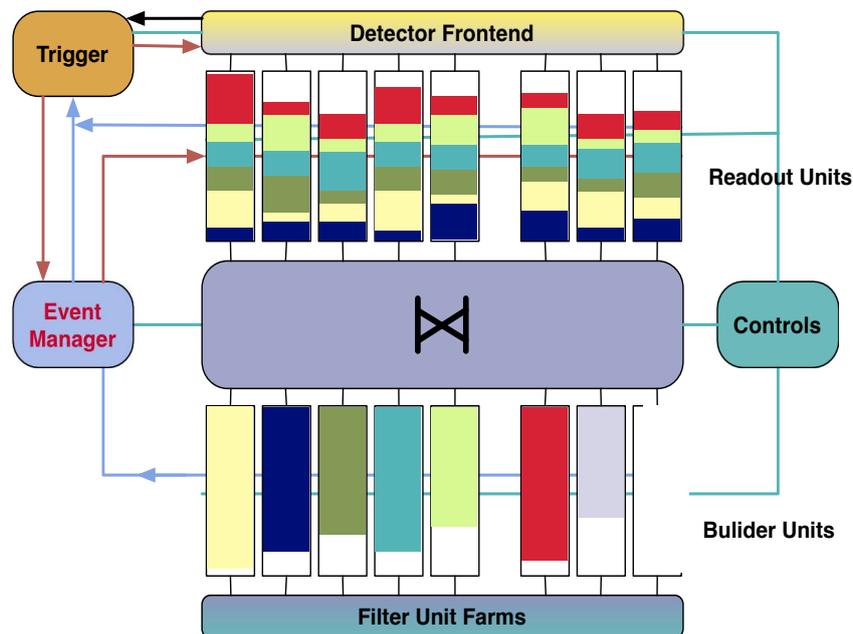


Figure 2. Schematic view of the readout builder (RU-builder): event fragments from the detector front-end readout modules are buffered in the readout units (RUs). The fragments are transferred using a large switching fabric to builder units (BUs) where they are assembled into complete events. The event manager (EVM) orchestrates the event building.

an event either rejecting it or sending it to the SM to be recorded, it tells the BU to discard it (7). When N events have been discarded, the BU returns the N tokens corresponding to the discarded events to the EVM (8), and the cycle continues.

2.4. XDAQ Framework

The RU-builder and the storage manager applications use the XDAQ framework. XDAQ is middle ware that eases the development of distributed data acquisition systems. It provides services for data transport, configuration, monitoring, and error reporting. The framework has been developed at CERN and builds upon industrial standards, open protocols and libraries [5].

3. Storing Data on Disk – The Storage Manager

The storage manager (SM) is located at the end of the online data chain. It receives events accepted by the high-level trigger (HLT) running on the filter units (FUs) (see figure 4). The accepted events are split over several I2O binary messages (fragments) and send over a dedicated Gigabit-Ethernet network to the SM. Each DAQ slice has one or two SM applications. The SM reassembles the fragments into complete events and writes them to disk. Events are stored in one or multiple files, depending on the triggers that fired for the event (the trigger bits) and the HLT process that selected the event (the trigger path). These files are buffered on a local NexSan SATABeast disk array (RAID-6) and subsequently copied to the central computing center (Tier 0), which repacks the events into larger files. These files are then fed to the offline event-reconstruction farms.

A subset of the events is sent to online consumers, using a HTTP request/response loop. Consumers either connect directly to one SM, or they connect to a proxy server which aggregates

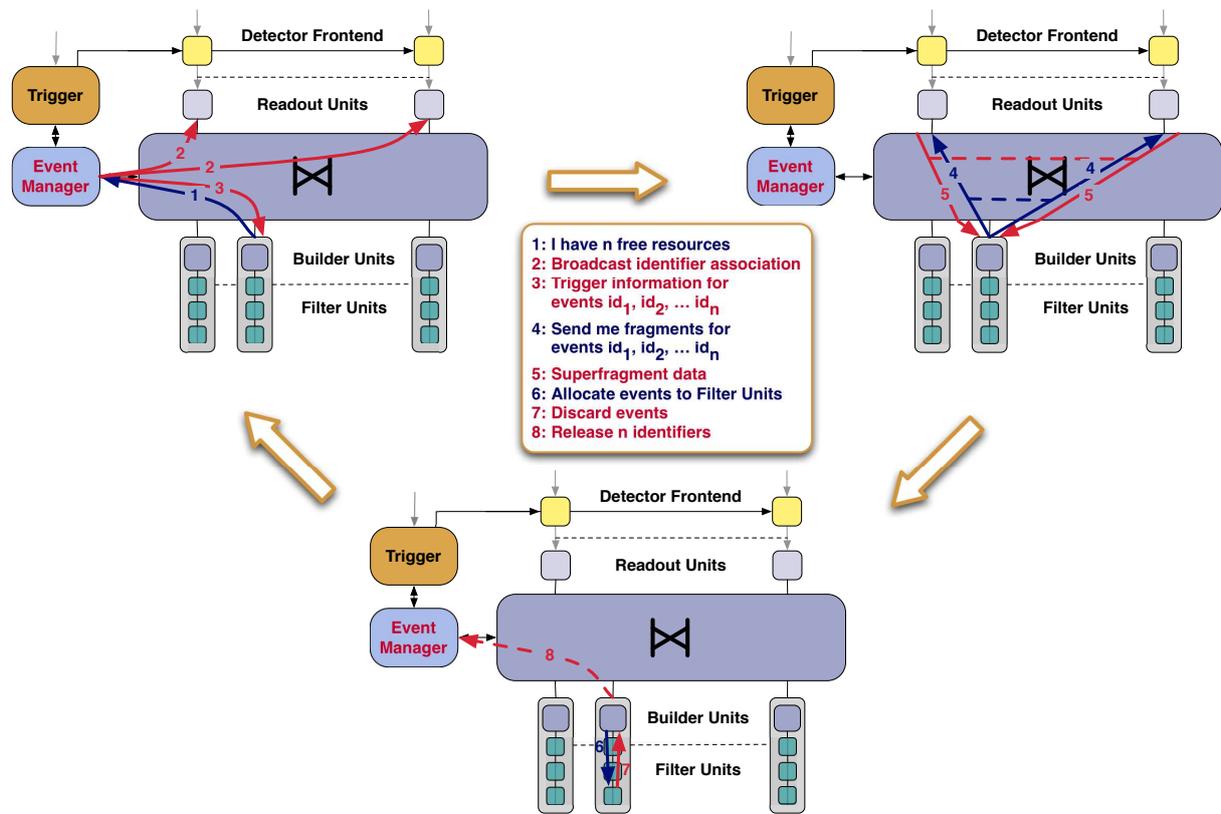


Figure 3. The event-building protocol uses a token-passing scheme (see text).

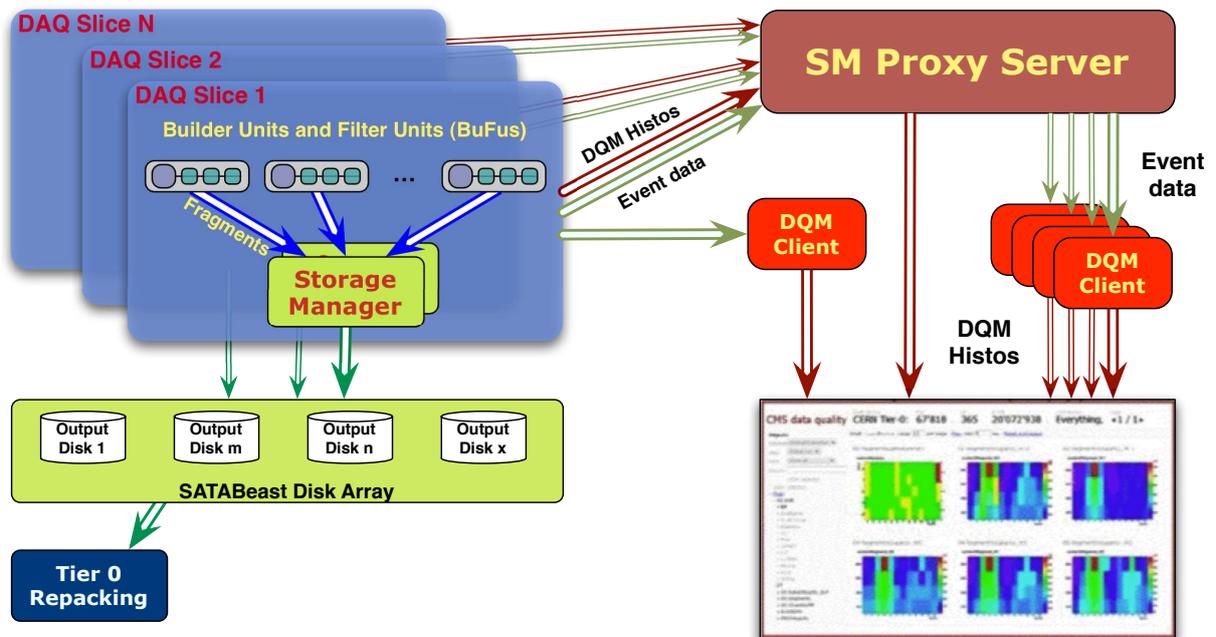


Figure 4. The storage manager writes accepted events to disk. It serves a subset of the events and data-quality (DQM) histograms either directly or via a proxy server to online consumers.

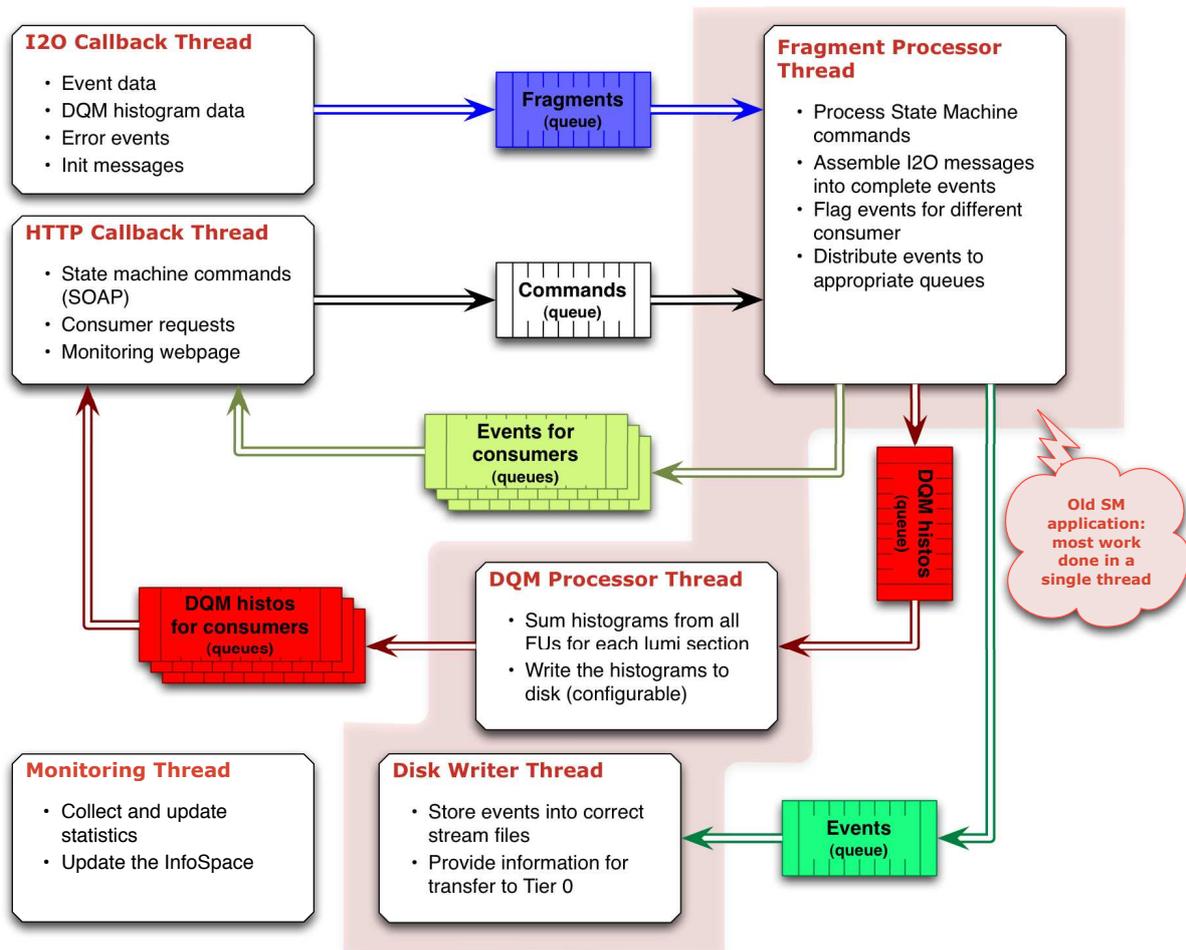


Figure 5. The storage manager application being redesigned to improve the robustness and performance by moving from a mostly single-threaded design to a multithreaded one. The figure shows the new thread model (white boxes) and the queues used to communicate between them. The shaded purple box shows the tasks performed in a single thread by the old SM application.

the data from all SMs in all DAQ slices. The consumers use this data for online data quality monitoring (DQM), for calibration purposes, or for displaying the events.

In addition, the FUs produce a set of histograms to monitor the data quality and trigger efficiencies. These histograms include events which are only available prior or during the event selection. These histograms are sent to the SM at each luminosity section boundary (every 93 seconds). The SM sums the histograms that it received from all FUs in the slice. It forwards the summed histograms to the proxy server. The proxy server receives the histograms from all SMs in all DAQ slices and sums them up again. The CMS data quality application retrieves the summed set of histograms for monitoring.

3.1. Redesign of the Storage Manager Application

The current implementation of the storage manager (SM) exposed some weaknesses during commissioning runs of the entire CMS detector recording cosmic rays. The main issue is that most of the work is done in a single thread. This caused dead-time in the system each time the

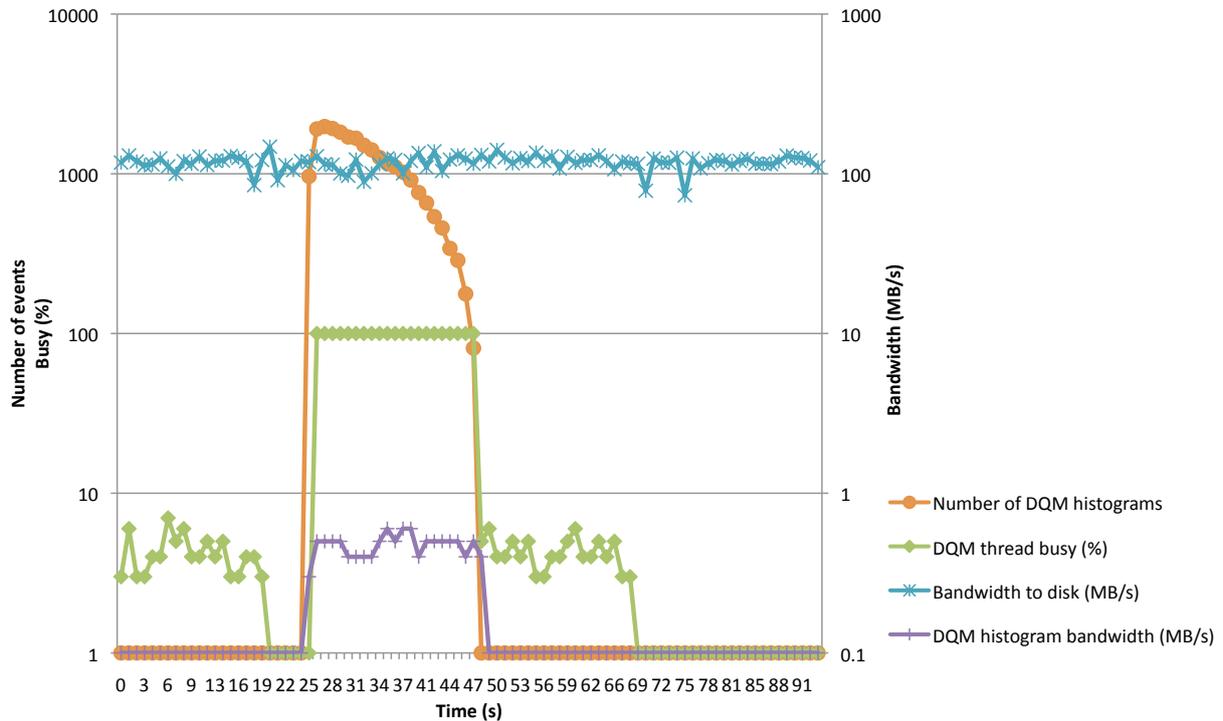


Figure 6. The performance of the redesigned storage manager application is shown at a moderate load. The bandwidth to disk is stable around 100 MB/s. At some point all DQM histograms are received, causing the number of DQM histograms in the queue to increase rapidly. The previously idle DQM thread becomes 100% busy and processes the DQM histograms for about 23s at a steady rate of 0.5 MB/s. Once all histograms are served to the consumers, the thread becomes idle again. This pattern repeats itself for each luminosity section (93 s).

SM was summing the DQM histograms for each luminosity section, i.e. every 93 seconds. In addition, the code evolved over several years and had been adapted to changing requirements. This resulted in a code base that was difficult to understand and hard to maintain. Therefore, the opportunity of the delayed LHC start-up was taken for a redesign and re-factoring of the existing code.

The new design (see figure 5) uses individual threads for the different tasks of the storage manager. The events are pushed into separate queues which are handled by dedicated threads. Care is taken that the main task of receiving events from the high-level triggers and writing them to the appropriate files (streams) is not blocked by the CPU-intensive task of summing DQM histograms, or by rather unpredictable requests from consumers of events or DQM histograms.

Figure 6 shows how the redesigned storage manager behaves under a moderate load as experienced during cosmic-ray data taking. It can be seen that the summing of the DQM histograms takes several seconds and takes 100% CPU time. The disk writing handled by another thread is not affected. In the previous, single-threaded design, the summing stalled the disk writing causing back-pressure on the upstream DAQ system.

4. Hardware

Figure 7 shows the location and the interconnection of the various hardware components used for the data-acquisition system.

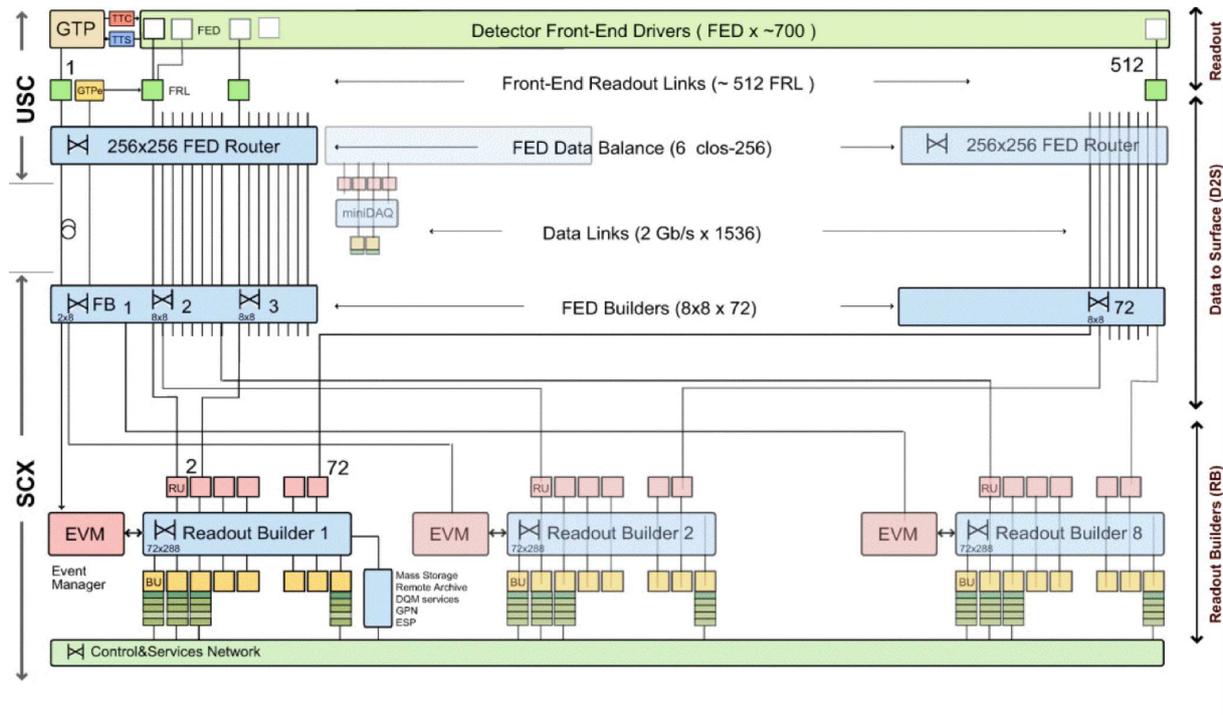


Figure 7. Schema of the hardware used for the data-acquisition.

The heart of the FED builder is a set of six Myrinet switches installed underground, close to the detector. The data is transported to the surface using 1536 optical links operating at 2 Gb/s. Another set of 72 8×8 Myrinet switches is located close to the event builder and high-level trigger (HLT) filter farm.

The RU-builder switching-fabric consist of 8 Force-10 E1200 switches [6] (one for each DAQ slice). Each switch has 6 90-port linecards with each port supports 11 Gb/s duplex. Each port is 2 times oversubscribed. The cabling is done to have full throughput on all ports using the fact that the traffic is basically uni-directional.

The RU-builder and HLT farm consist of 640 (32 racks) Dell PE 2850 dual dual-core, 2 GHz, 4 GB memory PCs used as RUs and 720 (24 racks) Dell PE 1950 dual quad-core, 2.6 GHz, 16 GB memory PCs used as BUs and filter units.

A Force 10 Gigabit Ethernet switch connects the storage manager to the HLT processors. A separate Gigabit Ethernet switch is used for transfer to Tier 0. The storage manager's hardware consists of 2 racks each with 8 SM nodes, 2 Fibre Channel switches (QLogic SanBox 5600) providing redundant pathways with automatic fall-over, and 4 NexSan SATABeasts (RAID-6 disk arrays). The latter provide a data buffer of 300 TB, which is equivalent of several days of data taking.

5. Operational Experience

The readout system has been successfully used to commission the detector, to collect more than 370×10^6 cosmic ray events, and to record the first LHC beam events in September 2008.

The event builder performs to the design specifications of 100 GB/s and the system has proven to be very flexible. The event builder can be subdivided into multiple independent readout systems that can be concurrently used to readout subsets of detector components. The system can be configured to focus on performance and reliability issues related to



Figure 8. 72 crates holding 8×8 Myrinet switches used for the FED builder located close to the event builder and high-level trigger farm.

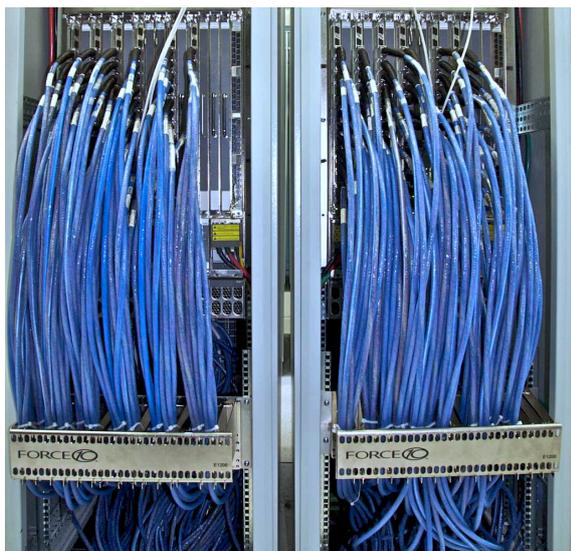


Figure 9. 2 of the 8 Force-10 E1200 switches used by the RU-builder.



Figure 10. Racks of Dell PE 2850 and PE 1950 computers used for the event building and for the high-level triggers.



Figure 11. Storage manager nodes connected through Fibre Channel switches (QLogic SanBox 5600) to the NexSan SA-TABeasts (RAID-6 disk arrays).

- high data throughput
- large number of input channels
- high CPU power in the high-level trigger
- high data rate to local disk.

In addition, data payloads can be injected at various points in the system to test its functionality and performance.

The 3-month long running of the experiment in fall 2008 highlighted several areas of improvement:

- The event builder needs to be tolerant against missing or corrupted front-end data.
- The error and alarm handling should be based on a uniform and system-wide approach.
- The Storage Manager application should handle CPU-intensive tasks in independent threads to assure that they do not block the readout (see section 3.1).

These improvements are being deployed and tested in the full system and will be available for the first physics run in fall 2009.

6. Conclusions

We discussed the architecture of the CMS two-stage event-builder and the storage of accepted events. The system has proven to be scalable and to out-perform the requirements to read-out the complete CMS detector. A few areas for improvement have been identified and are being addressed in time for the restart of LHC beam operations planned for the fall of 2009.

References

- [1] CMS Collaboration CERN-LHCC-2002-026
- [2] Myricom, see <http://www.myri.com>
- [3] Sakulin H *et al.* 2008 *IEEE Trans. Nucl. Sci.* **55** 190–197
- [4] Pieri M *et al.* 2006 Prepared for International Conference on Computing in High Energy and Nuclear Physics (CHEP 2006), Mumbai, Maharashtra, India, 13-17 Feb 2006
- [5] Gutleber J *et al.* 2009 Prepared for International Conference on Computing in High Energy and Nuclear Physics (CHEP 2009), Prague, Czech Republic, 21-27 Mar 2009
- [6] Force10, see <http://www.force10networks.com>