

The CMS High Level Trigger System: Experience and Future Development

G Bauer⁶, U Behrens¹, M Bowen², J Branson⁴, S Bukowiec², S Cittolin⁴, J A Coarasa², C Deldicque², M Dobson², A Dupont², S Erhan³, A Flossdorf¹, D Gigi², F Glege², R Gomez-Reino², C Hartl², J Hegeman², A Holzner⁴, Y L Hwong², L Masetti², F Meijers², E Meschi², R K Mommsen⁵, V O'Dell⁵, L Orsini², C Paus⁶, A Petrucci², M Pieri⁴, G Polese², A Racz², O Raginel⁶, H Sakulin², M Sani⁴, C Schwick², D Shpakov⁵, S Simon², A C Spataru², K Sumorok⁶

¹ DESY, Hamburg, Germany; ² CERN, Geneva, Switzerland; ³ University of California, Los Angeles, Los Angeles, California, USA; ⁴ University of California, San Diego, San Diego, California, USA; ⁵ FNAL, Chicago, Illinois, USA; ⁶ Massachusetts Institute of Technology, Cambridge, Massachusetts, USA

E-mail: Andrei.Cristian.Spataru@cern.ch

Abstract. The CMS experiment at the LHC features a two-level trigger system. Events accepted by the first level trigger, at a maximum rate of 100 kHz, are read out by the Data Acquisition system (DAQ), and subsequently assembled in memory in a farm of computers running a software high-level trigger (HLT), which selects interesting events for offline storage and analysis at a rate of order few hundred Hz. The HLT algorithms consist of sequences of offline-style reconstruction and filtering modules, executed on a farm of O(10000) CPU cores built from commodity hardware. Experience from the operation of the HLT system in the collider run 2010/2011 is reported. The current architecture of the CMS HLT, its integration with the CMS reconstruction framework and the CMS DAQ, are discussed in the light of future development. The possible short- and medium-term evolution of the HLT software infrastructure to support extensions of the HLT computing power, and to address remaining performance and maintenance issues, are discussed.

1. Introduction

The CMS [1] trigger and data acquisition system [2] is designed to cope with unprecedented luminosities and interaction rates. At the LHC design luminosity of $10^{34} \text{cm}^{-2} \text{s}^{-1}$, and bunch-crossing rates of 40 MHz, an average of about 20 to 40 interactions take place at each bunch crossing. The trigger system must reduce the bunch-crossing rate to a final output rate of O(100) Hz, consistent with an archival storage capability of O(100) MB/s. Only two trigger levels are employed in CMS: the Level-1 Trigger (L1T), implemented using custom electronics reduces the initial event rate by a factor of 100 [3] using custom electronics. Events accepted by the Level-1 are read-out and assembled by the DAQ Event Builder (EVB) [4]. The second trigger level, the High Level Trigger (HLT) analyzes complete CMS events at the Level-1 accept rate of 100 kHz. The HLT provides further rate reduction by analyzing full-granularity detector data, using software reconstruction and filtering algorithms on a large computing cluster consisting of commercial processors, the Event Filter Farm. In this paper we

describe recent experience with the CMS HLT during collision runs, as well as ongoing and planned development of the system.

2. Trigger and DAQ General Architecture

Due to the large number of channels and the short nominal interbunch time of the LHC (25 ns), only a limited portion of the detector information from the calorimeters and the muon chambers is used by the L1T system to perform the first event selection, while the full granularity data are stored in the detector front-end electronics modules, waiting for the L1T decision (Fig. 1). The overall latency to deliver the trigger signal (L1A) is set by the depth of the front-end pipelines and corresponds to 128 bunch crossings. The L1T processing elements compute the physics candidates (muons, jets, etc.) based on which the final decision is taken. The latter is the result of the logical OR of a list of bits (up to 128), each corresponding to a selection algorithm. All the trigger electronics, and in particular the set of selection algorithms, are fully programmable [3].

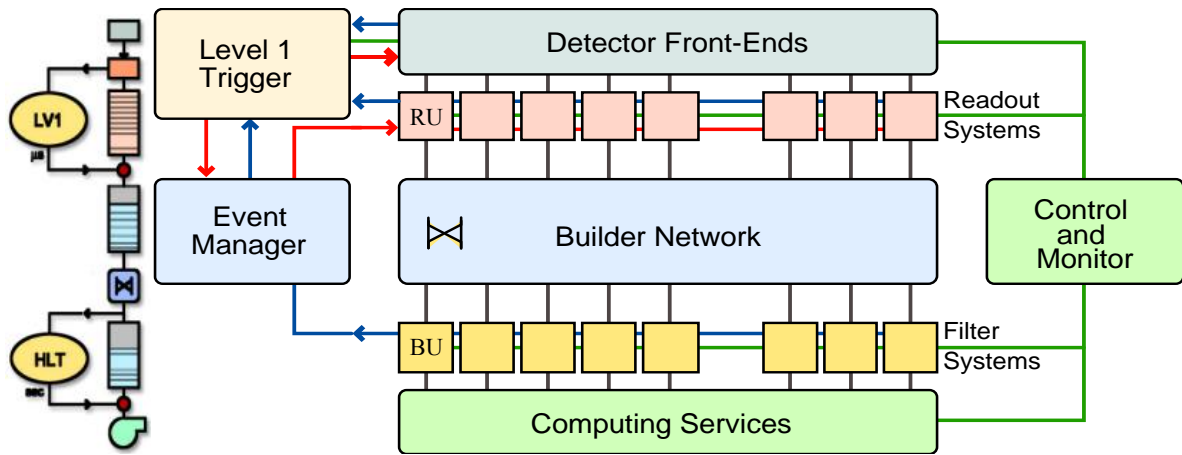


Figure 1. Schematic architecture of the CMS DAQ and Trigger System

Data fragments corresponding to events accepted by the L1T are read out from the front end modules and assembled into “super fragments” in a first stage of event building that uses Myrinet switches. They are then delivered to the Readout Units (RU). Builder Units (BU’s) receive super-fragments from the RUs via a large switch fabric based on Gigabit Ethernet, and assemble them into complete events. An Event Manager (EVM) provides the flow control by steering the event building based on trigger information. The two-stage event building approach is described in detail in [4].

3. High Level Trigger Architecture

The second stage of event building, assembling full events into the memory of the BU, is organized in slices, built around a monolithic GE switch. Triggers are assigned to each of the 8 independent slices in a round robin fashion, and each of the slices operates independently from the others. This principle is illustrated in Fig. 2.

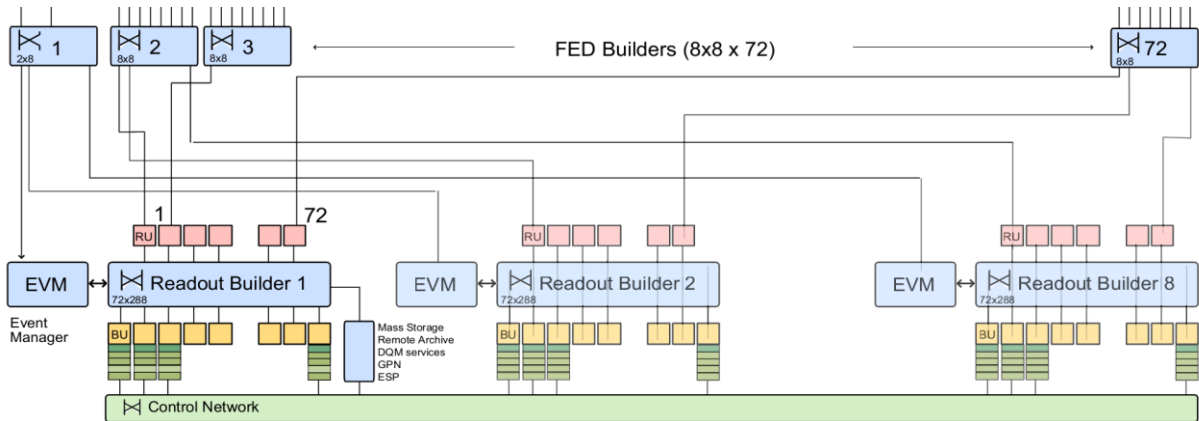


Figure 2. Event Builder and HLT nodes are arranged in slices. Components of the HLT are shown below the Readout Builder networks in each slice.

Events are pre-assembled in the Builder Unit / Filter Unit processor memory. An independent process, running the physics algorithms, subsequently analyzes each event. Accepted events are forwarded to the Storage Manager System (SM) for storage in a large disk pool. Stored events are transferred over a redundant 10 Gb fiber optic connection running in the LHC tunnel to the CERN computer center, where they are processed for analysis and archived in a mass storage system.

The complex of the BU/FU processing nodes and the SM form a large distributed computing cluster, the Event Filter Farm. Around 1000 rack-mounted commodity processors, connected to the Readout Builder by one or two GE connections, run the physics reconstruction and selection algorithms. The BUFU software structure is sketched in Fig. 3.

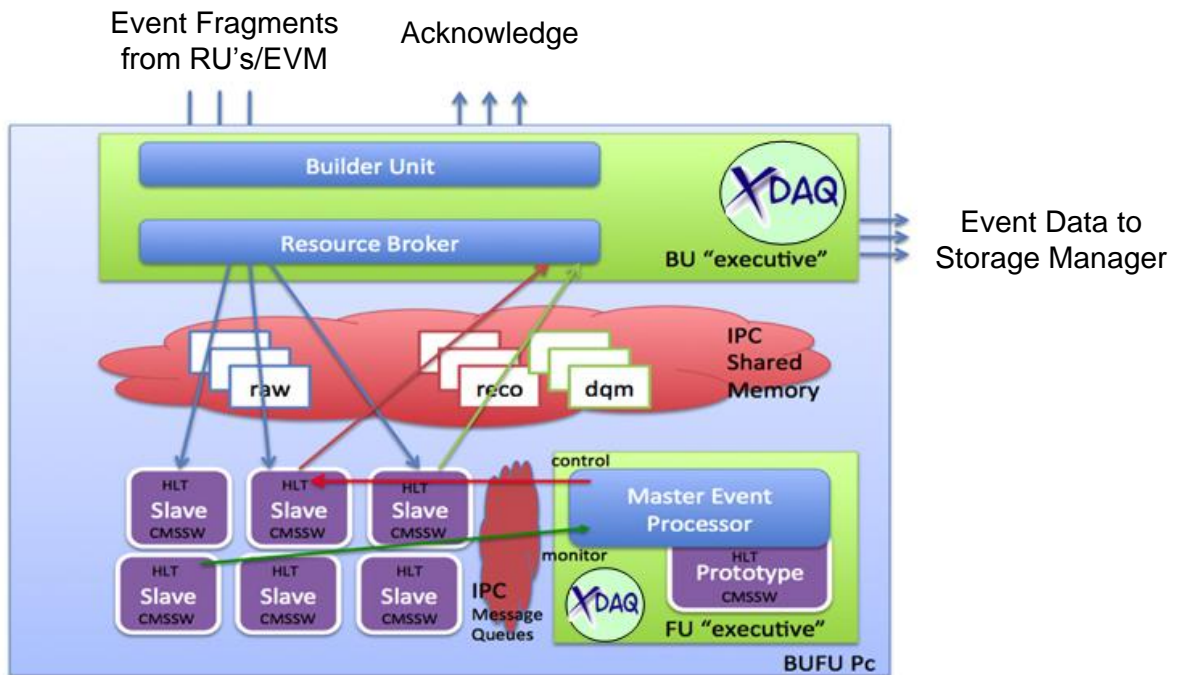


Figure 3. Block scheme of the BU/FU software

A Resource Broker (RB) requests events for processing from the BU and hands the corresponding data to slave Event Processors (EP). These processes are forked by the master Event Processor in order to fully utilize the number of available cores on each node. Since physics algorithms are resource-intensive and dependent on the nature of event data, they are decoupled from data flow by running in separate processes. Selected events are handed back to the RB over the same IPC structure, along with data quality information. The RB transfers them to the Storage Managers over the same switched network used for event building. Additional information on the system is available in [5] and [6].

The CMS Run Control and Monitoring System (RCMS) is charged with controlling the HLT components by employing a hierarchical structure of finite state machines that define the state of the DAQ. Built using Java web technologies, it allows quick recovery from problems and is optimized for efficiency. More details on the run control framework and architecture are available in [7].

4. Operation of the CMS HLT in 2010 and 2011

Operation of the CMS High Level Trigger was efficient during the LHC physics runs of 2010 and 2011. Due to the robustness and flexibility of the entire DAQ infrastructure, it was possible to adiabatically increase the CPU power of the HLT farm by deploying more machines with multi-core processors capable of handling the increasingly complex algorithms needed to recognize important events for physics.

Overall CMS data taking efficiency was 91% in 2011 (Fig. 4). The central DAQ system achieved a high availability percentage during stable beam periods of the LHC, reaching 99.7% in 2011. Most of the problems were caused by software, and were generally fixed as soon as identified. These issues could not be foreseen prior to deployment and running, since they were caused by changing operational conditions. When dealing with failed software or hardware in the Filter Farm, a new system configuration has to be loaded, excluding the problematic components. In order to speed up and simplify this task for the on-call experts, the system has been recently integrated with the DAQ Doctor expert system. With the help of the expert system, a new configuration can be generated in around 40 seconds.

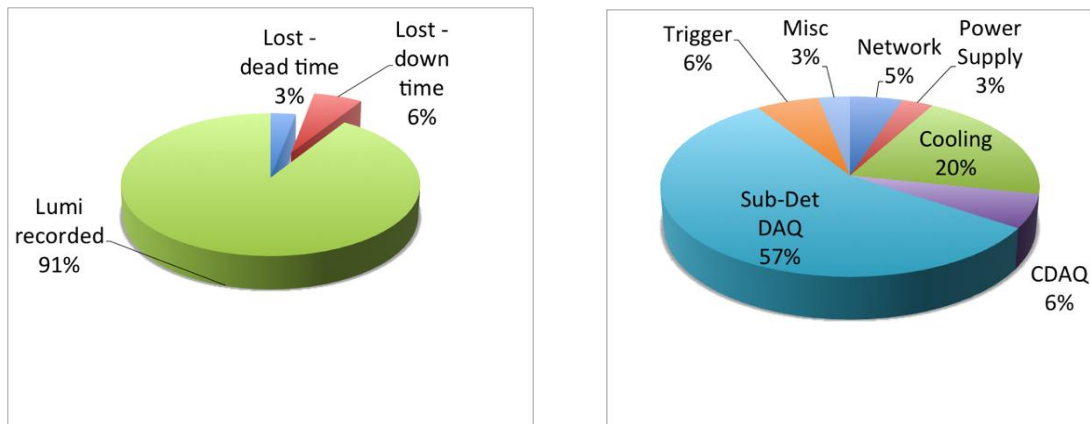


Figure 4. Overall CMS data taking efficiency in 2011; left chart shows total luminosity recorded by the experiment, right chart provides a breakdown of causes for downtime.

The DAQ system's high availability is also due to fault tolerant features. In the event of a crashing data flow node or software component, data taking continues. In the filter units, physics algorithms, sensitive to data quality and detector conditions, may occasionally crash or process input for a long time, in which case they are analyzed online using a dedicated data stream. The input data are stored for further analysis. These processes are then replaced by new ones during the ongoing run.

4.1. Hardware additions in the Filter Farm

The original HLT System of 720 units totaling 5760 cores was first extended in May 2011 with 72 units (3456 cores and hyper-threading capability), and then in May 2012 with a further 64 units (4096 cores and hyper-threading capability) (Table 1). The current HLT Filter Farm size is 13200 cores, allowing for a per-event CPU budget of around 150 ms/event at a rate of 100 kHz.

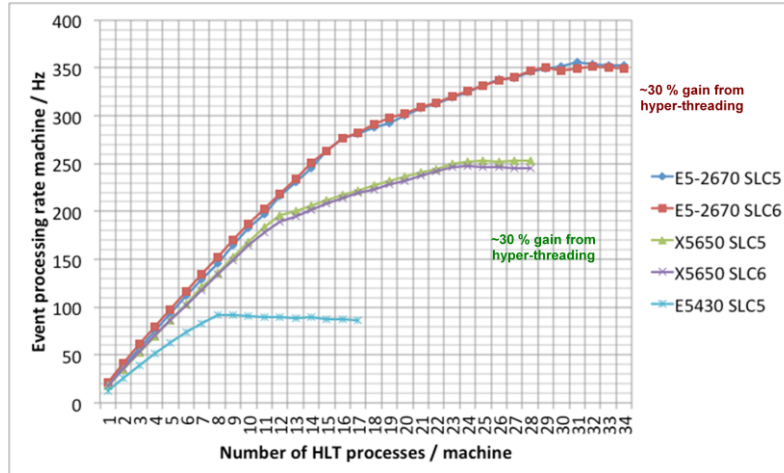


Figure 5. HLT machine performance; the three generations of machines present in the farm are evaluated, running either Scientific Linux CERN version 5 or 6 (SLC5-6)

Fig. 5 shows the event processing rate per machine in the High-Level Trigger farm as a function of the number of processes running on each one. The 8-core machines level-off at one process per core, while the recently added nodes also benefit from a 30% gain due to hyper-threading.

Table 1. Evolution of the Filter Farm hardware.

	<i>Original HLT System</i>	<i>2011 Extension</i>	<i>2012 Extension</i>
Form factor	1 motherboard in 1U box	4 motherboards in 2U box	4 motherboards in 2U box
CPUs per motherboard	2x 4-core, 2.66 GHz, 16GB RAM	2x 6-core, 2.66 GHz, hyper-threading, 24 GB RAM	2x 8-core, 2.6 GHz, hyper threading, 32 GB RAM
#boxes	720	72 (=288 motherboards)	64 (=256 motherboards)
#cores	5760	3456 (+ hyper-threading)	4096 (+ hyper-threading)
Cumulative #cores	5.6k	9.1k	13.2k
Cumulative #HLT processes	5k	11k	20k

5. Future development

The continuous extension process of the CMS HLT includes operations on both the software and hardware of the filter farm (as shown in Fig. 6). In order to cope with a factor of 2 expected increase in the luminosity delivered to the detector in 2012, new hardware has been installed to accommodate running more complex algorithms and higher selectivity of events. With higher luminosity also comes higher pile-up and thus more time consuming tracking, confirming the necessity of more processing power in the farm. The recent deployment of new hardware conforms to the CMS DAQ strategy of buying processing power just in time.

Software improvements are aimed at long term maintainability, comprising both refactoring and redesigning operations for software components. To this end, state model consolidation in the components is ongoing, while improvements of the inter-process communication methods between data flow and algorithm processes are being investigated. Another planned development is the further decoupling of physics algorithms and data flow processes, which are based on different software frameworks.

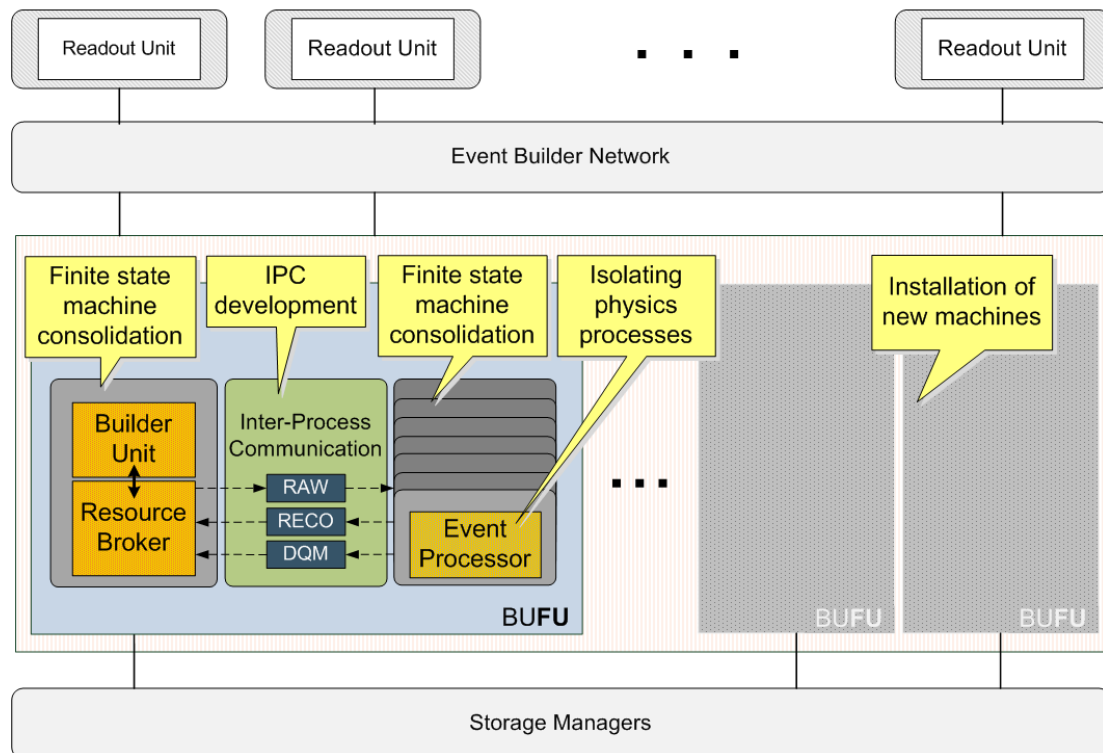


Figure 6. Future development of the CMS High Level Trigger System

5.1. HLT state models

After a period of evolution and adaptation, consolidation of the state models for HLT applications is currently underway, with the aim of improving robustness and ease of maintenance over the lifetime of the experiment. The RB application was recently refactored having replaced the previous state machine implementation with one using the Boost Statechart library [8]. A significant improvement brought on by the use of this library is having state-local storage, resulting in a clear separation of concerns between different states of the application. The code becomes easier to maintain, as well as document, since classes are correlated to UML semantics.

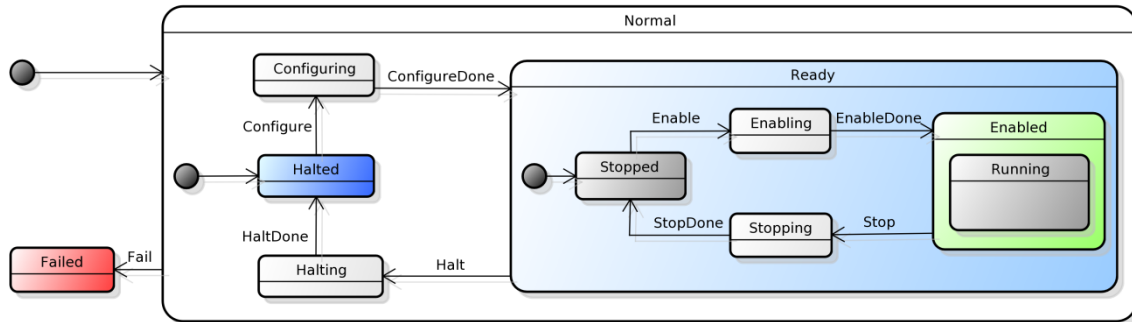


Figure 7. State machine diagram implemented in the recently refactored Resource Broker

The Statechart diagram in Fig.7 shows the states and transitions for the Resource Broker. The Boost library used in the implementation allows declaring states and transitions between them, as well as inner states. Having state-dependent behavior provides a clearly established status of the system by avoiding conditional branches. Leveraging this advantage, the callback functions for messages received from the Builder Unit are implemented only in those states that can handle the messages. For states that cannot handle messages, a minimal implementation of the callback is provided, logging information of the type of message received and possible implications.

By using inner states, reactions are easier to define and attribute to states. This is particularly useful when implementing behavior in case of failures. The Statechart framework will attempt to apply a transition to the current state and, if there is no reaction defined for it, will seek a reaction from all outer states. In case a *Fail* event occurs in *Running* state, reactions will be attempted in *Running*, *Enabled*, *Ready* and finally *Normal*, from which a transition to *Failed* state will be triggered.

Status reporting to RCMS is simplified when using this approach, by having a custom implementation of reporting actions in each state. An inner state such as *Running* does not need to be reported to RCMS, and there may be more inner states implemented in the system without external visibility. Instead, reporting to the control structure is done by those states that are relevant to the entire system, such as *Enabled*, *Ready*, *Halted*, *Failed*, or the “transitional” states *Configuring*, *Enabling*, *Stopping* and *Halting*.

The Boost Statechart Library is now used in the majority of the HLT software components with successful results. The remaining component to be refactored with this library is the Event Processor.

5.2. Inter-process communication in the HLT

The current method of data transfer between the event builder and physics algorithms in the Event Filter is a custom shared memory structure, comprising three types of shared memory cells: raw cells, reco cells and DQM cells. The RB obtains events from the BU and places them in raw cells, which are then read by EP’s running selection algorithms. Accepted events are placed in reco cells. They are picked up by the RB and sent to be stored by the SM. EP’s also generate Data Quality Monitoring (DQM) cells, which are also sent to the SM.

An alternative inter-process communication (IPC) method, message queues, is being investigated. By using message queues, process synchronization is delegated to the system, while the implementation of the IPC structure becomes simpler, by not having to handle synchronization and memory corruption prevention. Event data is exchanged between processes by posting and retrieving messages from the queue. The RB receives events from the BU, caches them locally and places them on the queue, as with raw memory cells. The cache is necessary on the RB side in order to ensure that no data is lost in case an EP application fails. EP’s retrieve raw messages from the queue, process them, and either place a reco message on the queue if the event is accepted, or simply instruct the RB to discard the event kept in local cache. The main benefits of this approach are the reduction in code complexity related to process synchronization, and a better design of the control mechanism for the

HLT selection and reconstruction processes. A potential downside of the message queue approach to event data transfer is a required extra copy operation, compared to the shared memory implementation. For an event to be sent over a message queue (e.g. from an RB to an EP), it has to be copied once onto the queue, and then once more from the message queue on the receiving side. Shared memory does not suffer this penalty, as a fragment copied by an RB onto a shared memory cell can then be directly read by an EP. However, this overhead is not expected to be problematic for the current memory capacity of the filter farm machines.

Another approach to IPC is a middle-ground one that combines the advantages offered by shared memory (design flexibility, high efficiency of access operations) with the ones provided by message queues (simpler to create custom protocols, synchronization handled internally, good point-to-point communication). High-volume data transfers can be handled by shared memory as in the current system, while control and time-sensitive messages to EP's are sent via message queues.

The recent refactoring of the RB resulted in an abstraction of the IPC method, allowing for different approaches to be implemented with relatively low impact.

5.3. Isolation of physics algorithms from system

The HLT physics selection and reconstruction algorithms have a strong connection with the nature of input data and detector conditions, and need frequent updates for both physics and CPU performance reasons. This is why these algorithms should become completely isolated from the rest of the DAQ system by moving them into standalone processes. In order to accomplish this, physics algorithms that are currently being run within the Event Processor will be moved to a separate process that is controlled by the EP. HLT algorithms could be run like batch processes, as in the case of the offline system. Expected benefits of such a change include an independent deployment cycle of HLT software as well as ensuring the stability of DAQ software dealing with data flow. This item is currently in the design and planning stage.

6. Summary

The CMS High Level Trigger System is in constant evolution in order to accommodate the increasing luminosities and interaction rates. This paper described the main directions of development of the HLT, both software and hardware, and gave an overview of the experience of recent runs with collision data. Section 2 provided a description of the entire CMS DAQ system architecture, while Section 3 focused on the High Level Trigger. In Section 4, recent operational experience with the HLT was noted, as well as a description of the hardware additions to the filter farm over the last few years. Finally, Section 5 gave an outline of the different directions for the software evolution in the farm, including consolidation of the state models, alternative approaches for inter-process communication and further separation of data flow from event processing components.

References

- [1] The CMS Collaboration, *The Compact Muon Solenoid Technical Proposal*, CERN LHCC 94-38, 1994.
- [2] The CMS Collaboration, CMS, *The TriDAS Project, Technical Design Report, Volume 2: Data Acquisition and High-Level Trigger*, CERN LHCC 2002-26, 2002.
- [3] Klabbers P. for the CMS Collaboration, *Operation and performance of the CMS Level-1 Trigger during 7 TeV Collisions*, Technology and Instrumentation in Particle Physics, 2011.
- [4] Bauer G. et al., *CMS DAQ Event Builder Based on Gigabit Ethernet*, IEEE Real Time, Batavia, IL, USA, 2007.
- [5] CMS Collaboration, *The CMS Experiment at the CERN LHC*, Journal of Instrumentation, Vol. 3, 2008.
- [6] The CMS High Level Trigger: Commissioning and First Operation with LHC Beams. Zanetti, Marco and Marta, Felcini. 2008, Proceedings of the IEEE Nuclear Science Symposium

- [7] Sakulin H. for the CMS Collaboration, *First operational experience with CMS Run Control System*, Detectors and Experimental Techniques, 2010.
- [8] Boost.org. *The Boost Statechart Library*. [Online] 2012.
http://www.boost.org/doc/libs/1_35_0/libs/statechart/doc/index.html

This work was supported in part by the DOE and NSF (USA) and the Marie Curie Program.