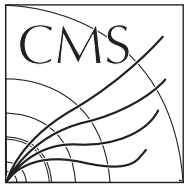
Available on CMS information server

CMS CR -2012/167



The Compact Muon Solenoid Experiment

Conference Report

Mailing address: CMS CERN, CH-1211 GENEVA 23, Switzerland

28 June 2012 (v3, 02 July 2012)

Recent experience and future evolution of the CMS High Level Trigger System

G Bauer[6)], U Behrens[1)], J Branson[4)] S Bukowiec[2)], O Chaze[2)], S Cittolin[4)], J A Coarasa[2)], C Deldicque[2)], M Dobson[2)], A Dupont[2)], S Erhan[3)], D Gigi[2)], F Glege[2)], R Gomez-Reino[2)], C Hartl[2)], A Holzner[4)], L Masetti[2)], F Meijers[2)], E Meschi[2)], R K Mommsen[5)], C Nunez-Barranco-Fernandez[2)], V O'Dell[5)], L Orsini[2)], C Paus[6)], A Petrucci[2)], M Pieri[4)], G Polese[2)], A Racz[2)], O Raginel[6)], H Sakulin[2)], M Sani[4)], C Schwick[2)], A C Spataru[2)], F Stoeckli[6)] K Sumorok[6)]

Abstract

The CMS experiment at the LHC uses a two-stage trigger system, with events flowing from the first level trigger at a rate of 100 kHz. These events are read out by the Data Acquisition system (DAQ), assembled in memory in a farm of computers, and finally fed into the high-level trigger (HLT) software running on the farm. The HLT software selects interesting events for offline storage and analysis at a rate of a few hundred Hz. The HLT algorithms consist of sequences of offline-style reconstruction and filtering modules, executed on a farm of 0(10000) CPU cores built from commodity hardware. Experience from the 2010-2011 collider run is detailed, as well as the current architecture of the CMS HLT, and its integration with the CMS reconstruction framework and CMS DAQ. The short- and medium-term evolution of the HLT software infrastructure is discussed, with future improvements aimed at supporting extensions of the HLT computing power, and addressing remaining performance and maintenance issues.

Presented at *RT2012: 18th IEEE NPSS Real Time Conference*

[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

Recent experience and future evolution of the CMS High Level Trigger System

Gerry Bauer, Ulf Behrens, James Branson, Sebastian Bukowiec, *Member*, *IEEE*, Olivier Chaze, Sergio Cittolin, Jose Antonio Coarasa, Christian Deldicque, Marc Dobson, Aymeric Dupont, Samim Erhan, Dominique Gigi, Frank Glege, Robert Gomez-Reino, Christian Hartl, Andre Holzner, Lorenzo Masetti, Frans Meijers, Emilio Meschi, Remigius K. Mommsen, Carlos Nunez-Barranco-Fernandez, Vivian O'Dell, Luciano Orsini, Christoph Paus, Andrea Petrucci, Marco Pieri, Giovanni Polese, Attila Racz, Olivier Raginel, Hannes Sakulin, *Member, IEEE*, Matteo Sani, Christoph Schwick , Andrei Cristian Spataru, Fabian Stoeckli and Konstanty Sumorok

***Abstract*–The CMS experiment at the LHC uses a two-stage trigger system, with events flowing from the first level trigger at a rate of 100 kHz. These events are read out by the Data Acquisition system (DAQ), assembled in memory in a farm of computers, and finally fed into the high-level trigger (HLT) software running on the farm. The HLT software selects interesting events for offline storage and analysis at a rate of a few hundred Hz. The HLT algorithms consist of sequences of offline-style reconstruction and filtering modules, executed on a farm of 0(10000) CPU cores built from commodity hardware. Experience from the 2010-2011 collider run is detailed, as well as the current architecture of the CMS HLT, and its integration with the CMS reconstruction framework and CMS DAQ. The short- and medium-term evolution of the HLT software infrastructure is discussed, with future improvements aimed at supporting extensions of the HLT computing power, and addressing remaining performance and maintenance issues.**

I. Introduction

THE CMS [1] trigger and data acquisition system [2] (Fig. 1) is designed to cope with unprecedented luminosities and interaction rates. At the LHC design luminosity of 10^{34}cm^{-2}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(500) Hz, consistent with an offline archival storage capability of a few hundred 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.

Manuscript received June 30, 2012. This work was supported in part by the DOE and NSF (USA) and the Marie Curie Program.

U. Behrens is with DESY, Hamburg, Germany.

S. Bukowiec, O. Chaze, J. A. Coarasa, C. Deldicque, M. Dobson, A. Dupont, D. Gigi, F. Glege, R. Gomez-Reino, C. Hartl, L. Masetti, F. Meijers, E. Meschi, C. Nunez-Barranco-Fernandez, L. Orsini, A. Petrucci, G. Polese, A. Racz, H. Sakulin, C. Schwick, and A. C. Spataru (corresponding author. phone: +41 22 76 62389, e- mail: Andrei.Cristian.Spataru@cern.ch) are with CERN, Geneva, Switzerland.

S. Erhan is with University of California, Los Angeles, California, USA.

J. Branson, S. Cittolin, A. Holzner, M. Pieri and M. Sani are with University of California San Diego, La Jolla, California, USA. S. Cittolin is also with Eidgenössische Technische Hochschule, Zurich, Switzerland.

R. K. Mommsen and V. O'Dell are with FNAL, Batavia, Illinois, USA.

G. Bauer, C. Paus, O. Raginel, F. Stoeckli and K. Sumorok are with Massachusetts Institute of Technology, Cambridge, Massachusetts, USA.

II. 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. 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].

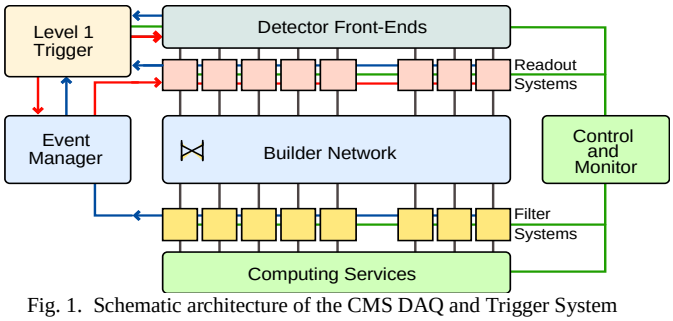


Fig. 1. Schematic architecture of the CMS DAQ and Trigger System

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].

III. 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.

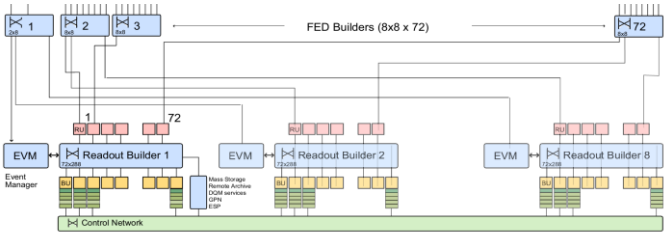


Fig. 2. Event Builder and HLT nodes are arranged in slices. Components of the HLT are shown below the Readout Builder networks in each slice.

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.

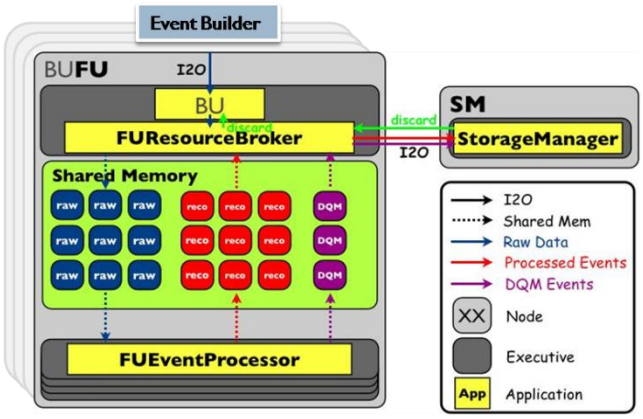


Fig. 3. Block scheme of the BU/FU software

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].

IV. Recent Experience with the CMS HLT

The CMS High Level Trigger operated efficiently during the LHC physics runs of 2010-2012. Thanks to the robustness and flexibility of the entire DAQ infrastructure, it was possible to adiabatically increase the CPU power of the (BU/FU) HLT farm by deploying more machines with multi-core processors, capable of handling the increasingly complex algorithms needed to recognize important events for physics.

HLT availability is monitored by the central DAQ system. During stable beam periods of the LHC, the HLT availability was 99.7% in 2011.

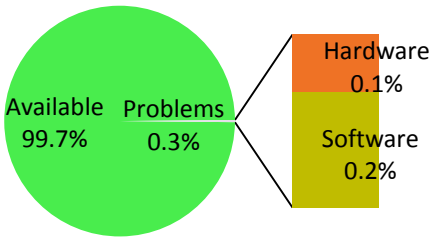


Fig. 4. Central DAQ availability in 2011 and breakdown of issues

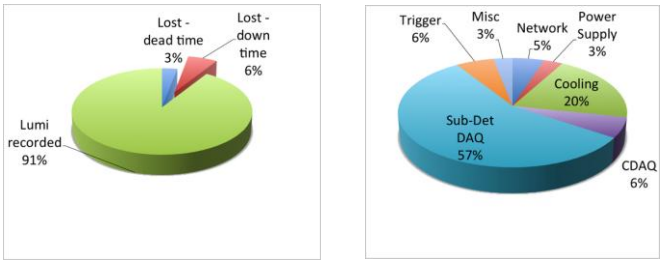


Fig. 5. Overall CMS data taking efficiency in 2011; left chart shows total luminosity recorded by the experiment, right chart provides a breakdown of causes for downtime.

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.

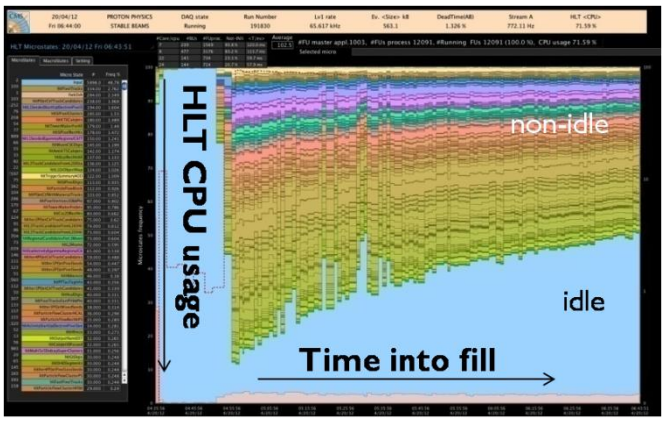


Fig. 6. CPU utilization in the Filter Farm at the start of a physics fill; data from April 2012.

Fig. 6 shows the CPU utilization in the HLT filter farm at the start of a physics fill. The state of every processing node is sampled every second. Colors on the graph indicate different HLT algorithms being run, while the light blue indicates idle time. CPU utilization reaches around 75% at the start of a physics fill, while the idle or "waiting for input" time presents a logarithmic evolution.

The high availability achieved by the system is also due to the fault tolerant design of the DAQ system. In case of a crashing data flow node or software component, data taking continues uninterrupted, with reduced throughput. In order to exclude failed components a new configuration must be loaded, implying a stop and start of the data taking run. Restarting a failed component without excluding it requires stopping and restarting a run. Due to the sensitivity of the physics algorithms to data quality and detector conditions, occasional crashes or long processing times occur. The problematic processes are analyzed online using a dedicated data stream, and the input data stored for further analysis. New processes are automatically started during the run to replace pathological ones.

V. Evolution of the HLT Infrastructure

The CMS HLT flexible design allows continued extensions of both the software and hardware of the filter farm. 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 event selectivity. With higher luminosity also comes higher pile-up and thus more time consuming tracking, requiring more processing power in the farm. The recent deployment of new hardware conforms to the CMS DAQ strategy of procuring and deploying processing power just in time.

Software improvements are aimed at long term maintainability, comprising both refactoring and redesigning operations of components. To this end, state model consolidation in the components is ongoing, while inter-process communication methods between data flow and algorithm processes, leveraging the progress in memory and processor I/O speed, are being investigated. Further decoupling of physics algorithms and data flow processes, which are based on different software frameworks, is being considered, to streamline the installation and update process and better exploit large multi-core architectures and memory buses.

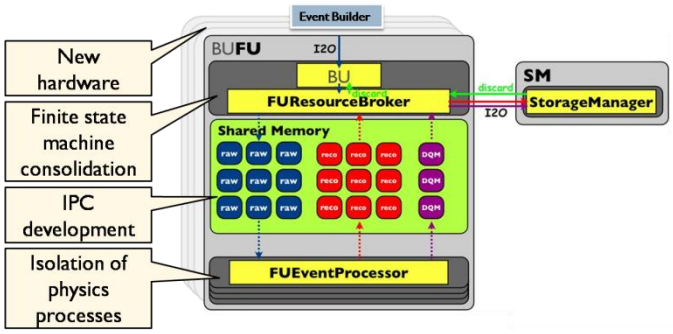


Fig. 7. Evolution of the HLT infrastructure.

A. 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, replacing 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 better definition of the system states and a lower coupling between them. The code becomes easier to maintain, as well as document, since classes are correlated to UML semantics.

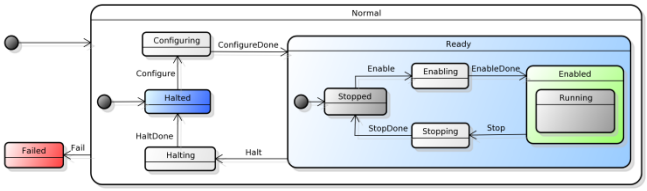


Fig. 8. State machine diagram implemented in the recently refactored Resource Broker.

The Statechart diagram in Fig. 8 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 and using the *State* design pattern. 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 by all but one application in the filter farm, the Event Processor, which is in line for the refactoring process.

B. Inter-Process Communication

The current method of data transfer between the data flow components and the 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 along with by-products of the selection process (e.g. tracks) which can ease event traceability. These are picked up by the RB and transferred to the SM for storage. EP's also generate Data Quality Monitoring data (DQM) cells, which are also sent to the SM.

Message queues are currently used as IPC method between master and slave Event Processors, for monitoring and control purposes.

The direction of development of the IPC method between data flow components and physics algorithms is to replace the current implementation with one based on message queues. In this way, drawbacks of shared memory, such as the ad-hoc inter-process synchronization, large number of semaphores (with potential for deadlocks) and high overall code complexity can be avoided. Advantages of using message queues include internal inter-process synchronization (thus reducing code complexity), and a simpler way to create custom communication protocols. Components thus become easier to maintain and extend.

TABLE I. IPC DEVELOPMENT STEPS FOR BUILDER UNIT – EVENT PROCESSOR COMMUNICATION (SM = SHARED MEMORY; MQ = MESSAGE QUEUES)

Data type	Current	Step 1	Step 2
Event	SM	SM	MQ
Control	SM	MQ	MQ

The first step in IPC development is to implement control data transfer over message queues. 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 second step implies event data being 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 are 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 issue a message on the data flow control queue to instruct the RB to discard the event kept in local cache.

In order to accommodate these modifications, the recent refactoring of the Resource Broker includes abstracting the IPC method in the implementation, thus reducing the impact of an eventual replacement of the IPC mechanism. A proof-of-concept implementation for the utilization of message queues for event data transfer has proved feasible in terms of performance.

C. Further Isolation of Physics Algorithms

There are two software frameworks currently used in the HLT: *XDAQ*[5] for online components and *CMSSW* [5] for selection and reconstruction algorithms. In order to execute event filtering, the *CMSSW* framework has to be loaded by the master Event Processor. The two frameworks have different software lifecycles, so there is scope to completely isolate them, by moving the selection and reconstruction algorithms to standalone processes (independent of *XDAQ*), controlled by the master EP. In this way, slave processes running algorithms would become unaware of DAQ processes dealing with data flow. HLT algorithms could then be run like batch processes in complete analogy to offline. The elimination of direct dependencies among the two frameworks would also allow decoupling the release and deployment cycles. This development is currently in the design and planning stage.

D. Deployment of New Hardware

The CMS Filter Farm, composed entirely of commercial hardware, is extended according to evolving computing power requirements. The latest extension was completed in May 2012. Higher luminosity implies a requirement for higher event selectivity and leads to more pile-up. As a consequence, more complex physics algorithms and more expensive tracking are required.

Fig. 9. Evolution of the HLT farm hardware; CPU models shown below units.

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).The current HLT Filter Farm size is 13200 cores, allowing for a per-event CPU budget of around 175 ms/event at a rate of 100 kHz.

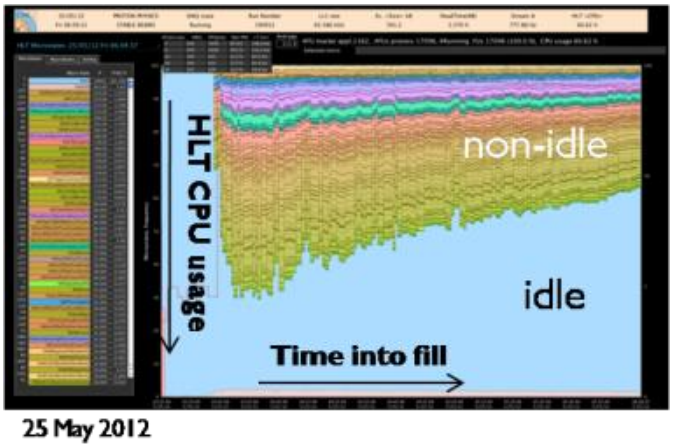


Fig. 10. HLT CPU utilization at the start of a physics fill after the installation of new hardware in May 2012.

As shown in Fig. 6, the HLT CPU was almost fully utilized at the start of a physics fill before the installation of new machines in the filter farm, with physics algorithms tuned to the available computing power. Fig. 10 shows a snapshot of the CPU utilization from May 2012. The accelerator performance in 2012 has increased rapidly, reaching peak luminosities of 6.5E33, more than double of 2011. The last extension has increased the available CPU time by another 50%, making the system capable of handling the even higher luminosity expected during 2012.

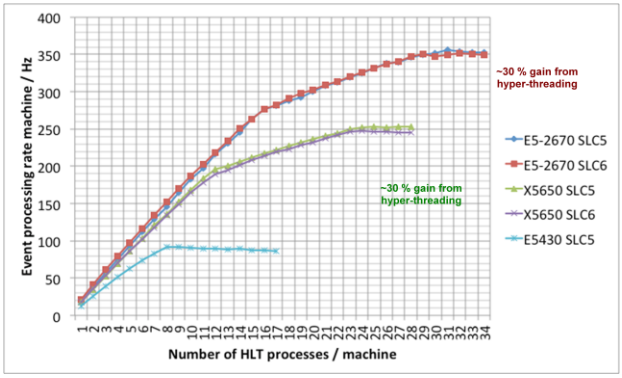


Fig. 11. HLT machine performance; the three generations of machines present in the farm are evaluated, running either Scientific Linux CERN version 5 or 6 (SLC5-6).

Fig. 11 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 nodes level-off at one process per core, while the recently added nodes (with 12 and 16 cores per node respectively) also benefit from a 30% gain due to hyper-threading.

VI. Summary

This paper first described up to date experience with the CMS High Level Trigger, which has been taking physics data with high efficiency during the past years. Then, the recent extension of the filter farm hardware was outlined. Finally, different directions of development for software components were detailed. The CMS High Level Trigger System is in constant evolution in order to accommodate the increasing luminosities and interaction rates delivered by the LHC.

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] CMS Collaboration, "Commissioning of the CMS High Level Trigger*",Journal of Instrumentation*, vol. 4, no. 10, 2009.

[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