Journal of Physics: Conference Series **513** (2014) 012006 doi:10.1088/1742-6596/513/1/012006

Deferred High Level Trigger in LHCb: A Boost to CPU Resource Utilization

M. Frank, C. Gaspar, E. v. Herwijnen, B. Jost, N. Neufeld
CERN, 1211 Geneva 23, Switzerland

E-mail: Markus.Frank@cern.ch

Abstract. The LHCb experiment at the LHC accelerator at CERN collects collisions of particle bunches at 40 MHz. After a first level of hardware trigger with output of 1 MHz, the physically interesting collisions are selected by running dedicated trigger algorithms in the High Level Trigger (HLT) computing farm. This farm consists of up to roughly 25000 CPU cores in roughly 1600 physical nodes each equipped with at least 1 TB of local storage space. This work describes the architecture to treble the available CPU power of the HLT farm given that the LHC collider in previous years delivered stable physics beams about 30% of the time. The gain is achieved by splitting the event selection process in two, a first stage reducing the data taken during stable beams and buffering the preselected particle collisions locally. A second processing stage running constantly at lower priority will then finalize the event filtering process and benefits fully from the time when LHC does not deliver stable beams e.g. while preparing a new physics fill or during periods used for machine development.

1. Introduction

LHCb is a dedicated B-physics experiment at the LHC collider at CERN [1]. The LHCb detector was designed to record proton-proton collisions at a rate up to 40 MHz delivered by LHC at a center of mass energy up to 14 TeV. LHCb exploits the finite lifetime and large mass of charmed and beauty hadrons to distinguish heavy flavor particles from the background in inelastic pp scattering. The first level trigger (Level 0), which reduces the rate of accepted events to 1 MHz, is hardware based. The second level or High Level Trigger (HLT) is purely software based. As shown in Figure 1, the readout boards (Trigger ELectronics Level 1 board/TELL1) send data from particle collisions at a rate of roughly 1 MHz through a switching network to the HLT farm nodes, where dedicated algorithms compute the decision on whether the event is to be accepted. Events with a positive decision are sent to the storage system and subsequently to the LHCb computing grid portal [2] for later offline analysis. The HLT hardware consists of roughly 1600 dual processor sockets grouped to 56 sub-farms, which host about 45000 trigger processes. These processes execute on heterogeneous worker nodes with 16 to 32 cores which are equipped with roughly 2 GB of memory per core. The local disk is used to host cached software. Accepted events to be kept for later offline analysis are sent to the storage system, where the data streams get partially replicated and sent to the monitoring facilities, the Monitoring and the Reconstruction farm.

The experiment controls system (ECS) handles the configuration, monitoring and operation of all equipment. All macroscopic entities of the ECS are modeled as Finite State Machine (FSM) entities [3], grouped together in a tree structure. The state of every higher level node summarizes

the state of its children. The FSM tree mechanism is used to describe the functioning of the processor farms such as the HLT processor farm, where at the lowest level processes are modeled as FSM elements, a set of processes is grouped to a node, a set of nodes describes a sub-farm and finally the set of sub-farms represents the high level trigger. All computing elements are shared and several concurrent instances may be mapped to the HLT farm or parts thereof.
During the 2012 running period the LHC collider delivered stable beams to be used for collecting particle collisions typically for about 30% of the time, as shown in Figure 2. The remaining time is necessary to ramp the magnets between two periods of stable beams, to improve the collider during machine development periods of several days or simply to solve technical problems. During these periods the CPU resources of the HLT farm are not required to operate the LHCb experiment.

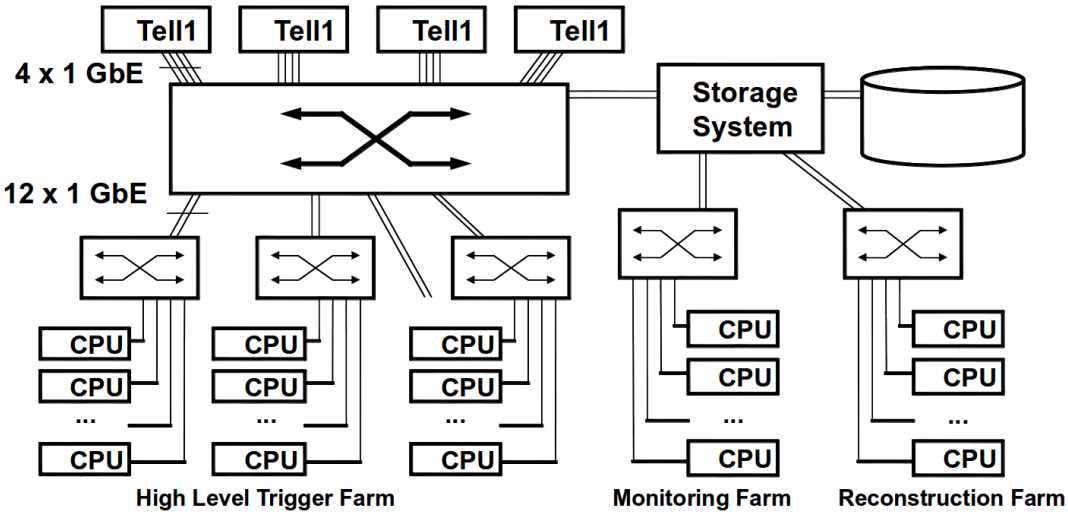

Figure 1. The layout of the LHCb DAQ hardware. Data from particle collisions are sent from the front-end boards (Tell1) through a switching network to the workers of the HLT farm (CPU).

In the following sections we present the mechanism to take advantage of the periods without particle collisions and to fully exploit the CPU resources of the HLT farm. The concepts and the configuration of individual nodes is described in section 3, the data monitoring and the controls aspects concerning the operation are discussed in section 4.

2. Problem Analysis and Solution

HLT operation is typically synchronous with the delivery of the particle collisions: it starts once the beams are declared stable and ends when the beams are dumped. During this period the front-end boards send massive amounts of data resulting from the detector response to the HLT farm. There the data must be processed since only a small fraction of the data may be kept for long term storage. The processing step is very CPU intensive:

- The front-end boards deliver L0-passed event data with a rate more than 50 GB/s, a rate too high for the long term storage system to cope with. In a farm of 1600 nodes this corresponds to an average input data rate per node of of roughly 30 MB/s.
- The worker nodes are equipped with a local disk with a capacity of 1 TB, suitable to intermediately buffer the arriving data, giving more than 1.5 PB of data buffer.
- During the 2012 data taking period about 25% of the L0-passed events were buffered to local disk and the HLT event selection applied later to these events. The result was a sizable improvement of 25%, still for large periods without beam, the HLT farm was idle.

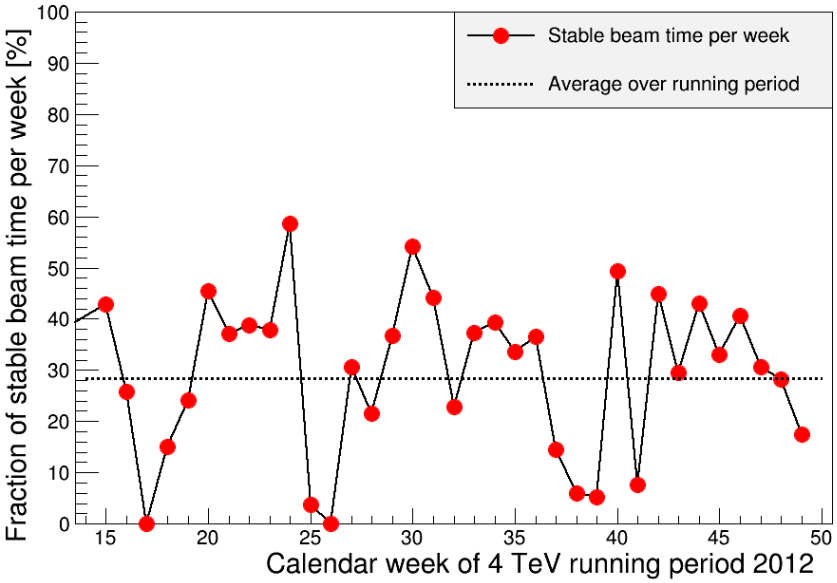

Figure 2. The LHC collider delivers beams useful to collect particle collisions during roughly 30% of the available time (data source: [4]). During this period the HLT farm must be operational to filter particle collisions. Otherwise the data is lost. The dips in the distribution indicate LHC machine development phases typically lasting more than a week. The average was computed over the entire period including machine development periods.

- The high level trigger program contains two major stages, HLT1 and HLT2. The program will be split into two independent asynchronous processes which gives further advantages due to the additional degree of freedom.
- Possible advantages arise from simple budget calculations. Parameters are the CPU consumption per event of HLT1 to be executed in real-time while taking data, the CPU consumption of HLT2 and the size of HLT1 accepted data stored on disk, since HLT2 requires more CPU resources and hence is slower than HLT1.
- Given the resources, this leads to a maximal execution time of HLT1 of 25 msec, a minimal buffer time of 11.5 days provided a HLT1 data reduction rate of six [5] and a LHC duty cycle of 30%. Hence, the maximum HLT2 budget would be 330 msec provided there is no organizational loss in switching activities.
- In order not to exhaust the buffer capacities in the local data buffers, the second trigger phase HLT2 is expected to constantly run on each worker node sending the finally accepted events to the long term storage, however, at a lower priority to not negatively interfere with the HLT1 processing.

In the following we describe the actual implementation of the above plan. The basic building blocks to achieve the required flexibility are described as well as their final composition and the control of these components by the experiment's control system.

3. Architectural Concepts and Implementation

The independent processing of two event filter stages required a substantial change of the flow of event data in the experiment [6], which was realized with rather limited manpower by the consequent reuse of a often reoccurring pattern. This pattern and its usage within the worker nodes is described in the following sections.

3.1. The Buffer Manager Concept

In the HLT every readout function is separated into independent tasks running asynchronously in order to de-randomize the flow of events. Readout functions are for example the assembly of events from fragments send by the TELL1 end boards, the event filtering or sending of accepted events to the long term storage. This allows to easily re-use the basic building block for processing event data shown in Figure 3 [6]. The producer task is responsible for reading/assembling data from data sources. Once finished, the data block is declared to a managed shared memory area, the Buffer Manager (BM) and the producer is ready for the next read operation. The consumer task is activated each time an event is declared in the BM. The consumer is responsible for releasing the space occupied as soon as it finished processing. The two activities of reading and processing can proceed asynchronously, provided there is sufficient space in the buffer to accommodate at least one read operation by the producer. A BM is uniquely identified by a name. Any number of instances may exist on a given worker node. The BM is managed by a dedicated task executing on each node, which creates and initializes the shared memory area and handles the access requests of the various producer and consumer applications.

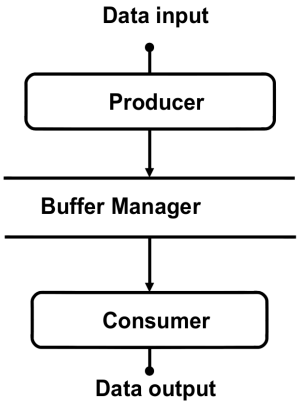

Figure 3. The schematic usage of the buffer manager concept as the basic building block for any event data processing within the data acquisition of the LHCb experiment.

3.2. Data Flow in the Worker Nodes

The buffer manager concept was consistently applied to achieve the required functionality. As shown in the left part of the dataflow diagram in Figure 4 data recorded by the experiment must be effectively reduced while the LHC beams are in physics settings and saved to the local disk. The data arrive in the various worker nodes from the front-end boards, get assembled by the *Eventbuilder* [6]. The assembled events are placed in a managed shared memory buffer and picked up by the first stage filter processes HLT1. The number of filter processes depends on the CPU resources of a particular node and is a configurable parameter. The accepted events are then put in an output buffer, from where the events are written to the local disk-buffer by the *DiskWriter*. A small fraction of the events with an integrated rate of 1 kHz or about 1 Hz per worker node, is sent for monitoring purposes to the monitoring facility to quickly spot possibly malfunctioning detector components. The first level data reduction is performed with high priority, events not accepted during this phase are lost for later offline physics analysis.

To not overflow the local disk buffer of the worker node (see Figure 4) the stored data is reduced by the continuously running second stage of the event filter process, HLT2, which executes more sophisticated algorithms and hence requires more CPU resources - though at lower priority. The HLT2 activity is independent from the actual data acquisition process. Though, with respect to the event data flow it is a parallel data taking activity - not with a detector as front-end, but

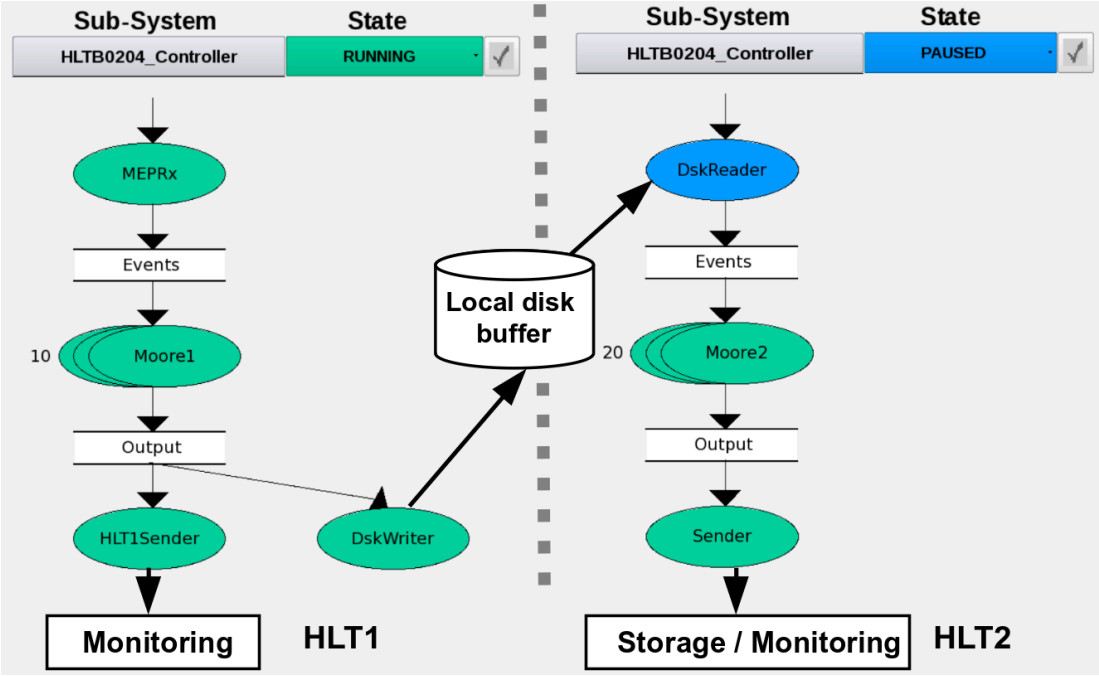

Figure 4. The event data flow in a worker node. The left part of the picture shows the flow of events to be preselected by the HLT1 processes (Moore1). The right side of the figure shows the secondary event selection by the HLT2 processes (Moore2). The events accepted are finally sent to the long term storage and stored there for later offline data analysis. The picture above consists of partial screen shots of the controls system.

events injected from disk. The preselected events are read from the local disk by a data reader process and placed in the *Events* buffer. The events are consumed by the second level event filter processes HLT2. All events accepted by this event filter are placed into the *Output* event buffer and sent to the storage system for offline analysis and data monitoring.
Figure 4 shows that the HLT1 is only connected to the HLT2 activity via the local disk as buffering device. Hence, both activities can be executed and controlled independently and the data taking process does not impose any fundamental restrictions on the execution of HLT2 unless the the local disk buffer is exhausted. If this happens, the event builders (MEPRx processes in Figure 4) will only be able to allocate space in the *Events* buffer at the rate the HLT2 processing can liberate local disk space. Though the system would not be at halt, this is clearly not a desired working point. To ensure flexibility, the two dataflows for HLT1 and HLT2 may be easily changed using a dedicated editor application. A palette of such configurations allows to quickly reconfigure the processes on each worker node.

3.3. Data Monitoring

The HLT1 and HLT2 activities are executed asynchronously in each node of the HLT farm. Each activity supplies a fraction of the accepted events to independent partitions [7] of the two monitoring facilities, the monitoring farm and the reconstruction farm as shown in Figure 1. Though the monitoring hardware is shared by the HLT1 and HLT2 activity, the data streams are kept separate while performing the following tasks:

- Events accepted by the HLT1 are used in a dedicated *Monitoring farm* for low-level monitoring to ensure the well functioning of the hardware components of the experiment and the operations aspects of the HLT1 selection process. A fraction of these events is

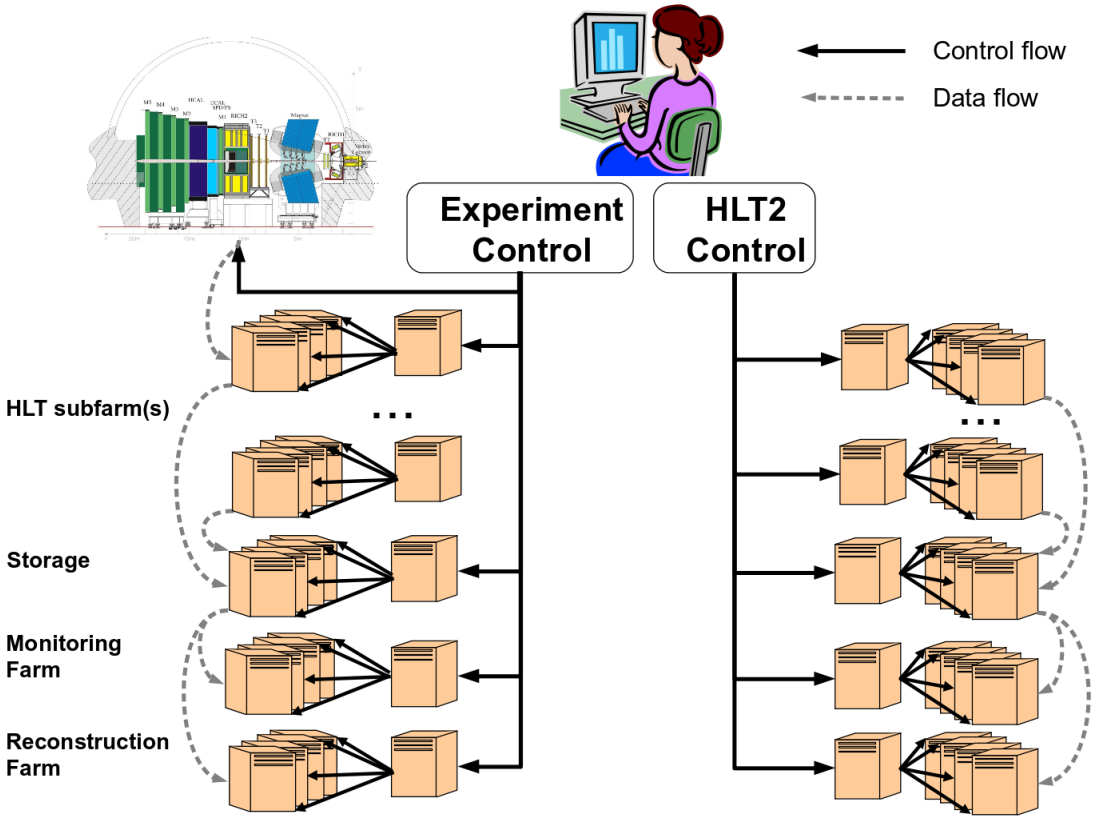

Figure 5. The overall experiment control with two distinct controls-trees: To the left the experiment control including the $HLT1$ event filtering and to the right the asynchronous second stage of HLT2 event filtering.

reconstructed at run-time in the *Reconstruction farm* for high-level monitoring using tracks and vertices.

- In order to enable the HLT2 to do a more offline like reconstruction, and hence better retain interesting events, the calibration is done using HLT1 accepted data so that up-to-date constants may be used in HLT2. These constants are determined in the *Monitoring farm* and the *Reconstruction farm*.
- Events accepted by the HLT2 selection phase are used to monitor the performance of the HLT2 processes and to verify the proper configuration of the software by checking physical distributions using selected physics channels. If the monitoring processes require reconstructed data, they execute in the Reconstruction Farm, else in the Monitoring Farm.

4. The Controls Hierarchy

The control of all participating equipment and all the processes, which belong to a given activity is performed by a tree-like hierarchy. For a given activity firstly all processes in an HLT node are grouped, then nodes are grouped to sub farms and finally sub farms to the HLT farm. The experiment components and all processes connected to the event filtering in the HLT farm, in the monitoring, the reconstruction and the storage cluster must be started in a well defined sequence to ensure that infrastructure dependencies such as buffer managers are present and network ports are open before clients try to use them. While taking data, there are only two such activities: As shown in Figure 5 to the left, for data taking the control of the experiment components together with the HLT1 event filter and the corresponding monitoring actions and on the right of Figure 5 the deferred HLT2 processing.

In parallel in a separate tree hierarchy, the HLT2 event filtering is continuously executed. From the controls aspect this tree is nearly identical to the full experiment control except, that no experiment hardware is manipulated and configured: At the highest level the operator issues commands to each activity, which is responsible for the overall orchestration of the state transitions. Both the Experiment Control and the HLT2 Control solely deal with macroscopic entities like units representing the HLT farm, the monitoring, reconstruction or storage partition. The HLT2 controls infrastructure ensures that no HLT2 event processing is started in the absence of proper calibration constants which are computed during the data taking of the HLT1 activity. The separation of the two controls trees was the result of the systematic application of the partitioning concept [8], which also allows subdetectors to independently perform data taking activities in periods without physics beam. The HLT2 is simply another data taking activity with no subdetector components attached.
The controls hierarchy was constructed using the commercial SCADA system WinCC OA [9], and SMI++ [10], a state manager interfaced to WinCC OA. All processes executing in the HLT farm, the monitoring, reconstruction and storage cluster are based on the LHCb data processing framework Gaudi [11], they accept transition requests from SMI++ and publish their state to SMI++ using the DIM protocol [12].

5. Conclusions

The buffer manager technique together with a flexible controls system supporting multiple parallel data acquisition streams allows to implement a very flexible event filter architecture. This architecture together with local event buffering on the worker nodes will help to extend the usable CPU time of the HLT farm for physics event filtering beyond the time the experiment actually will receive particle collisions from the LHC collider. After the current long shutdown LHCb will benefit in 2015 of a gain of CPU time useful for event filtering of a factor 3 depending on the collider performance. The gain will be achieved by separating the event filtering into a fast and a slower component executing asynchronously. In addition, the high level trigger decisions will be of higher quality using the run-time calibration- and alignment constants.

References

[1] LHCb Collaboration, "LHCb the Large Hadron Collider beauty experiment, reoptimised detector design and performance", CERN/LHCC 2003-030
[2] A. Tsaregorodtsev et al., "DIRAC: a community grid solution", 2008 J. Phys.: Conf. Ser. 119 062048.
[3] E. van Herwijnen, "Control and Monitoring of on-line trigger algorithms using Gaucho", International Conference on Computing in High Energy and Nuclear Physics (CHEP 2006), Mumbai, India, 2006; proceedings.
[4] LHC fill summary table, stable beam duration for 4 TeV running 2013, http://lhc-statistics.web.cern.ch/LHC-Statistics/index.php.
[5] J. Albrecht, G. Raven, V. Gligorov, private communication.
[6] M. Frank et al., "The LHCb High Level Trigger Infrastructure", International Conference on Computing in High Energy and Nuclear Physics (CHEP 2007), Victoria, BC, Canada, proceedings.
[7] O. Callot et al., "Data Monitoring in the LHCb Experiment", International Conference on Computing in High Energy and Nuclear Physics (CHEP 2007), Victoria, BC; proceedings.
[8] C. Gaspar et al., "Partitioning in LHCb", LHCb 2001-116 DAQ.
[9] SIMATIC WinCC Open Architecture [Online], http://www.pvss.com
[10] C. Gaspar et al., "SMI++ - Object Oriented Framework for Designing Control Systems for HEP Experiments", International Conference on Computing in High Energy and Nuclear Physics (CHEP 1997), Berlin, Germany, 1997; proceedings.
[11] G. Barrand et al, "GAUDI : The software architecture and framework for building LHCb data processing applications", Comput. Phys. Commun. 140 (2001) 45-55.
[12] C. Gaspar et al., "DIM, a Portable, Light Weight Package for Information Publishing, Data Transfer and Inter-process Communication", International Conference on Computing in High Energy and Nuclear Physics (CHEP 2000), Padova, Italy; proceedings.