ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THDPL01

CONFIGURING AND AUTOMATING AN LHC EXPERIMENT FOR FASTER AND BETTER PHYSICS OUTPUT

C. Gaspar, R. Aaij, F. Alessio, J. Barbosa, L. Cardoso, M. Frank, B. Jost, N. Neufeld, R. Schwemmer, CERN, Geneva, Switzerland

Abstract

LHCb has introduced a novel online detector alignment and calibration for LHC Run II. This strategy allows for better trigger efficiency, better data quality and direct physics analysis at the trigger output. This implies: running a first High Level Trigger (HLT) pass synchronously with data taking and buffering locally its output; use the data collected at the beginning of the fill, or on a run-by-run basis, to determine the new alignment and calibration constants; run a second HLT pass on the buffered data using the new constants. Operationally, it represented a challenge: it required running different activities concurrently in the farm, starting at different times and load balanced depending on the LHC state. However, these activities are now an integral part of LHCb's dataflow, seamlessly integrated in the Experiment Control System and completely automated under the supervision of LHCb's 'Big Brother'. In total, for all activities, there are usually around 60000 tasks running in the ~1600 nodes of the farm, and the load balancing of tasks between activities can be done within 1 second. In addition, if/when some CPU power is still available, an extra activity for Offline Simulation can also be started concurrently. The mechanisms for configuring, scheduling and synchronizing different activities on the farm and in the experiment in general will be discussed.

INTRODUCTION

LHCb [1] is one of the four experiments at the Large Hadron Collider (LHC) at CERN. Around the end of 1997, a common project, the Joint Controls Project (JCOP) [2], was setup between the four LHC experiments and a Controls group at CERN, to define a common architecture and a framework to be used by the experiments in order to build their Detector Control Systems (DCS).

The JCOP Framework [3] adopted a hierarchical and highly distributed architecture providing for the integration of the various components in a coherent and uniform manner. The Framework was implemented based on a SCADA (Supervisory Control And Data Acquisition) system called WinCC-OA (formerly PVSSII) [4]. While WinCC-OA offers most of the needed features to implement a large control system, it was felt that a tool for implementing higher-level logical behavior was missing. For this reason, the JCOP project was complemented by the integration of SMI++ [5]; a toolkit for sequencing and automating large distributed control systems, whose methodology combines three concepts: object orientation, Finite State Machines (FSM) and rule-based reasoning.

Unlike the other LHC experiments, LHCb decided to use the JCOP concepts and tools not only for the DCS but for all areas of control in the experiment. The aim was to achieve an integrated and coherent Experiment Control System (ECS) by using a common approach and the same tools and components throughout the system.

THE EXPERIMENT CONTROL SYSTEM

LHCb's ECS handles the configuration, monitoring and operation of all experimental equipment in all areas of the Online System:

- The Experiment's Infrastructure: magnet, cooling, electricity distribution, detector safety, etc.
- The Detector Control System (DCS): gases, high voltages, low voltages, temperatures, etc.
- The Data Acquisition System (DAQ): front-end electronics, readout network, storage etc.
- The Timing and Fast Control System (TFC): timing and trigger distribution electronics.
- The L0 Trigger (L0): the hardware trigger components.
- The High Level Trigger (HLT) Farm: thousands of trigger algorithms running on a large CPU farm.
- The Monitoring Farm: A smaller farm running monitoring tasks to produce histograms for checking online the quality of the data being acquired
- Interaction with the outside world: LHC Accelerator, CERN safety system, CERN technical services, etc.

The relationship between the ECS and the other online components of the experiment is shown schematically in Fig. 1. This figure shows that the ECS provides the unique interface between the operators and the experiment's equipment.

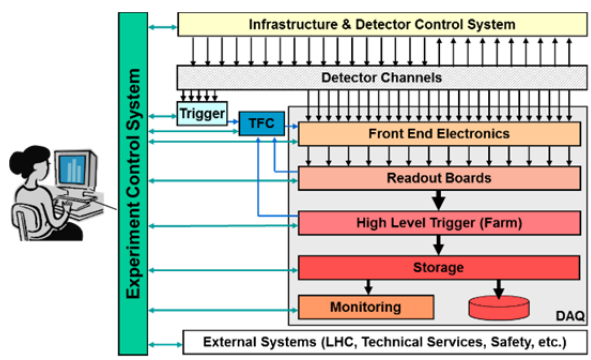

Figure 1: Scope of the Experiment Control System.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THDPL01

Due to the size and complexity of the experiment, LHCb adopted a hierarchical, highly distributed, tree-like, structure to represent the structure of sub-detectors, sub-systems and hardware components. This hierarchy allows a high degree of independence between components, for concurrent use during integration, test or calibration phases, but it also allows integrated control, both automated and user-driven, during physics data-taking. Figure 2 shows a simplified version of LHCb's control system architecture.

This architecture is composed of two types of building blocks: Control Units, which are logical entities capable or controlling and monitoring the sub-tree below them and Device Units, which represent real entities like a temperature sensor, a high voltage channel or a software task.

The control of the High Level Trigger farm is implemented as a sub-system (HLT in Fig. 2) and completely integrated in the Experiment Control System

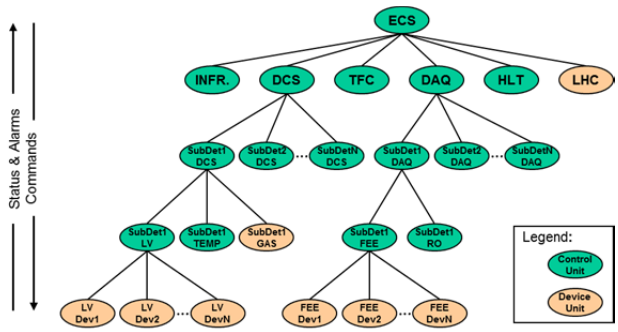

Figure 2: LHCb Simplified Architecture.

THE HIGH LEVEL TRIGGER

LHCb's High Level Trigger is in charge of selecting interesting events for physics analyses. The trigger algorithms run distributed on a large farm, composed of around 1600 computers containing around 50000 cores. The farm is organized in 62 sub-farms, each one with up to 32 nodes. Since the farm computers were bought or replaced at different times the farm is quite heterogeneous and the farm computers have been split into 4 categories (at the moment): Slow (24 cores), Medium (32 slow cores), Fast (32 cores) and Faster (40 cores). Figure 3 shows the context of the HLT farm within the Online System and the Data Acquisition's data flow.

From a software point-of-view, all tasks involved in the data acquisition's data flow, i.e. moving or processing data for various purposes (triggering, monitoring, storage, etc.), are based on the Gaudi Online Framework [6] (the offline framework complemented with online services providing hooks for control and monitoring) and respect the same pattern (centered around a buffer manager concept) as shown in Fig. 4.

The tasks running on the HLT farm are no exception and the simplest dataflow within a farm node is represented in Fig. 5.

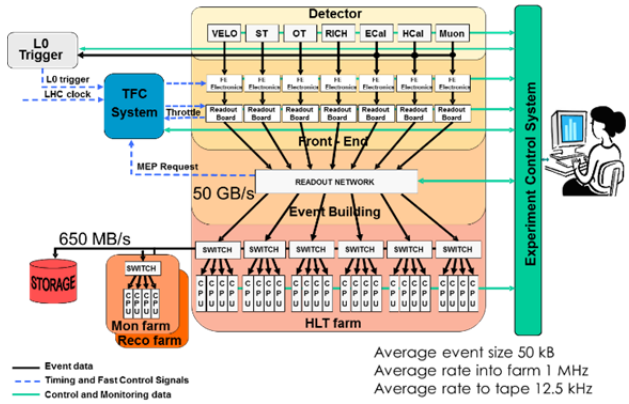

Figure 3: The HLT Farm in the Online System context.

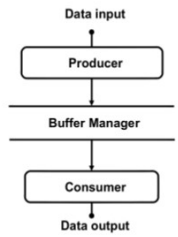

Figure 4: The Dataflow Software Pattern.

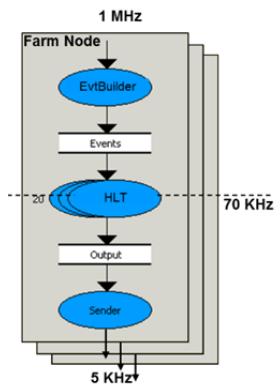

Figure 5: Old-Style HLT dataflow.

Until 2012 the HLT farm used this paradigm, but it was noticed that running the HLT synchronously with the data taking was a waste of CPU resources, since the farm works at full speed while the LHC delivers physics collisions but is then completely idle outside these periods. Knowing that the LHC delivers stable beams about 30% of the time and that the physicists were eager to use the available CPU power to improve their trigger capabilities, a first improvement was made to the system: The deferred HLT, represented in Fig. 6.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THDPL01

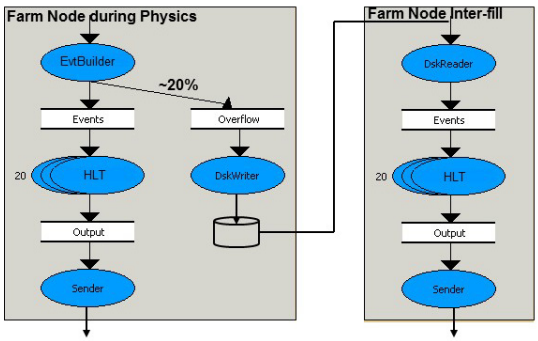

Figure 6: Deferred HLT dataflow.

In this new paradigm, 20% of the data was deferred to the local disk of each farm node and processed during the inter-fill gaps. After each fill, the farm was automatically stopped and reconfigured for inter-fill mode, with local disk input.

In 2015, for Run II, a major improvement was made to the HLT processing: The Split HLT, represented in Fig. 7. The idea was twofold: use even better the CPU and disk resources available and obtain offline quality data directly at the HLT output. For this reason the HLT was split into two separate tasks, HLT1 and HLT2 (this was already the case but they used to run sequentially within the same process). This allows storing the data on disk at the HLT1 output, which requires significantly less buffering space than at HLT input, and using the HLT1 output to perform detailed alignment and calibration, which is needed in HLT2 to obtain offline quality reconstruction.

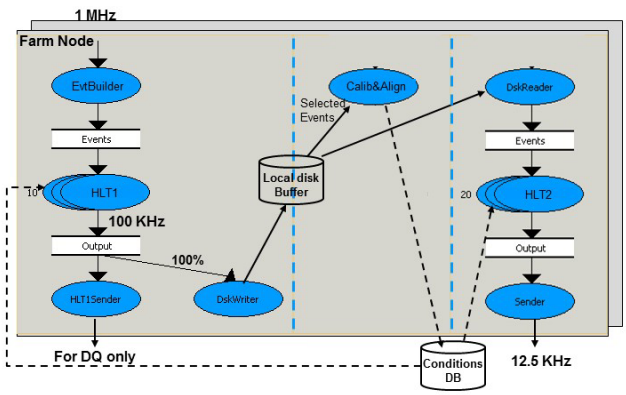

Figure 7: Split HLT dataflow.

This new HLT architecture requires three dataflow slices to run independently and asynchronously in each farm node:

- The first slice performing HLT1 and storing the data locally runs synchronously with physics data taking.
- The second slice involves normally a single task per node for alignment purposes and runs for a few minutes whenever sufficient data is available to perform a certain type of alignment.
- The third slice runs semi-permanently in the background. It performs HLT2 on the data previously acquired and stored locally, and finally sends the results out to central storage.

CALIBRATION AND ALIGNMENT

Two main types of tasks are scheduled to run between HLT1 and HLT2: calibration tasks and alignment tasks. Calibration tasks are normally standalone tasks, they use a subset of the HLT1 data and provide Calibration constants on a run by run basis (for example the RICH and OT subdetectors calibration). Alignment tasks are normally iterative procedures and need significant CPU resources, so they run distributed over the whole HLT farm. Alignment procedures are normally run on a fill-by fill basis (examples are the VELO, Tracker, MUON and Rich Mirror alignment). Both types of tasks produce new constants that the HLT2 processing must wait for in order to process the respective data. But some constants are so important that they need to be taken into account immediately, thus when they change they trigger a "Run Change" in the Run Control so that also HLT1 can pick them up.

Figure 8 shows (on the right) the list of groups of constants that the HLT2 must wait for in order to process each run.

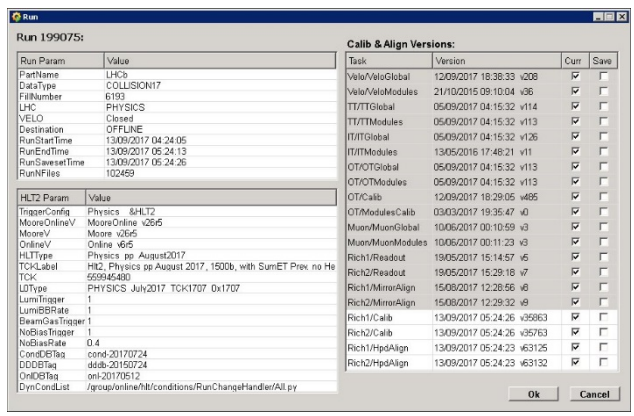

Figure 8: Parameters to be used for processing a certain Run in HLT2.

THE HLT FARM CONTROL SYSTEM

The original HLT farm control system was conceptually quite simple: a tree structure composed of sub-farms, containing nodes, which contained processes. With the Split HLT we need three independent control trees representing the three dataflow slices: The first to run HLT1, synchronous with data taking, a second one to run the Alignment tasks and a third to run HLT2. The changes to the control system tree are depicted in Fig. 9.

A second modification visible in Fig. 9 is the composition, in terms of children, of a Farm Node: the task composition was "hardwired" inside each farm node, now the type of tasks running depend on the "slice" they belong to. In the new system the concept of "Dataflow Architecture" was introduced.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THDPL01

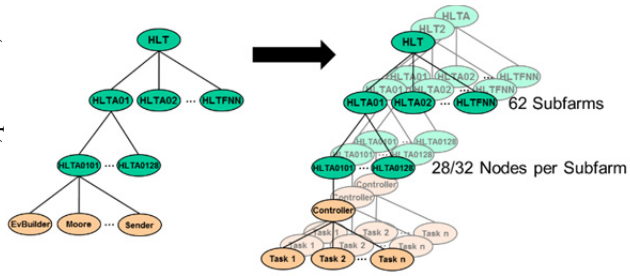

Figure 9: Changes to HLT Control System.

The Dataflow Architecture Concept

The Dataflow Architecture defines the tasks that need to run on a certain node for a certain purpose, this definition is read at configuration time by a controller process in the relevant machine. This concept is used not only in farm nodes but also in other machines in the data acquisition dataflow that need to start tasks. The Dataflow Architectures are edited graphically, an example is shown in Fig. 10.

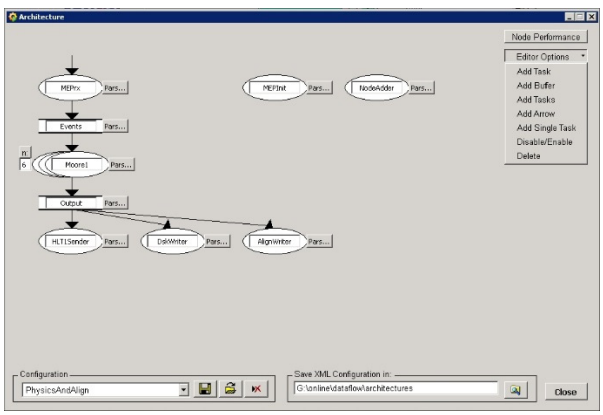

Figure 10: Architecture example (HLT1).

A nice side-effect of this editor, is that the graphics can, and are, then used at run-time to show the status of the tasks in a certain node, as the example in Fig. 11 shows.

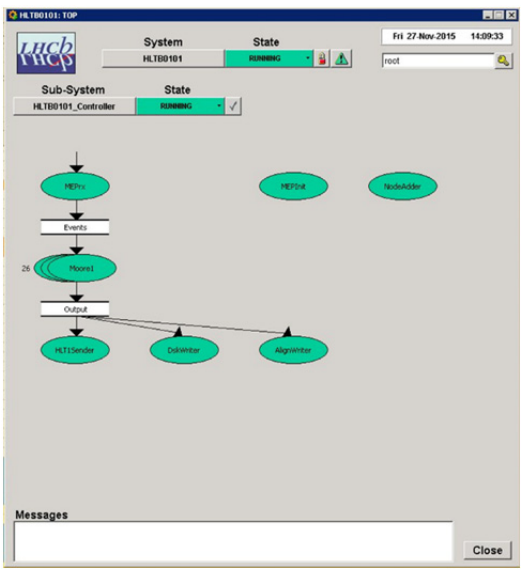

Figure 11: Architecture used at Run-Time (HLT1).

In an initial phase the number of processing tasks for example for HLT1 or HLT2 was defined in the Architecture, this turned out to be problematic, since it required a re-configuration of the full farm to change the number of tasks for a given slice.

The Dataflow Scenario Concept

"Dataflow Scenarios" were introduced in order to dynamically (and automatically) change the number of processing tasks on the nodes depending on the slice, LHCb activity and node type. The general idea is that during PHYSICS collisions we need enough HLT1 tasks to cope with the data rate without inducing dead-time and still running HLT2 in the background; while outside PHYSICS (normally called End-of-fill periods) we can lower the number of HLT1 tasks to 1 per node and maximize the number of HLT2 tasks to process as fast as possible and empty the local buffers as quickly as possible. A third scenario is applied at RAMP, which happens just before PHYSICS is declared to start increasing the number of HLT1 tasks in preparation of Physics events arriving. Figure 12 shows the current End-of-fill (EOF) Scenario settings.

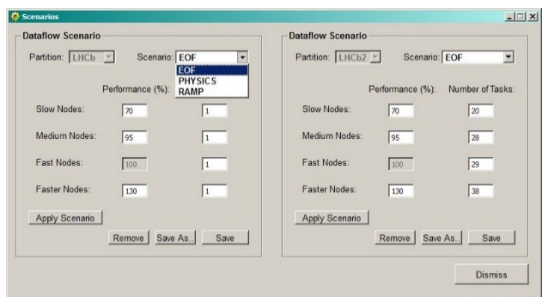

Figure 12: Dataflow Scenario for End-Of-Fill.

The Scenarios are automatically applied when the LHC changes its operational state and within a second the number of tasks for the different slices gets adjusted. The controller takes care of spawning previously configured tasks, so they are immediately operational, when the number increases and stopping extra ones when the number decreases. Figure 13 shows the graphic of HLT1 (Moore1) tasks and HLT2 (Moore2) tasks during a period of one day where the LHC delivered several fills.

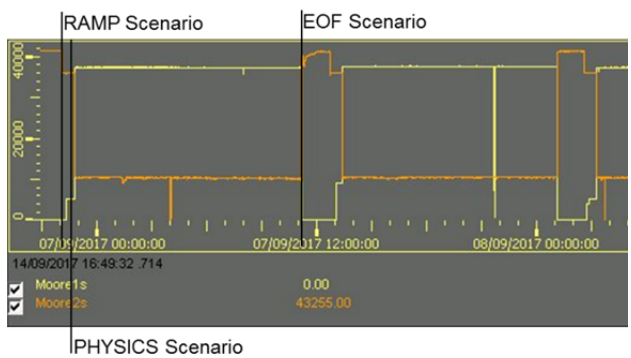

Figure 13: HLT1 and HLT2 number of tasks.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THDPL01

HLT FARM AUTOMATION

Like all sub-systems, the Farm Control is implemented using the JCOP Framework and in particular the FSM toolkit. Even though the tree structure is quite similar for the three farm slices, the logic each one implements can be quite different. In general the three slices have in common automatic procedures to exclude broken farm nodes while in operations and also to try to include them back when fixed. The rest of the logic is different:

- In the HLT1 slice all nodes are configured and started or stopped synchronously, i.e. all nodes at the same time.
- In the HLT2 slice each node behaves autonomously. Each node has a little auto pilot that checks when there are local files to be processed and passes them on to the trigger processes together with the appropriate conditions for each run.
- The Alignment slice performs an iterative procedure. At top-level an iterator process is started and one process in each node responds to commands from the iterator and reports results back, until the iterator decides that the procedure has converged. In this slice excluded nodes are not brought back automatically as they would interfere with the iteration results.

RUN CONTROL AUTOMATION

As described before, the Experiment Control System is in charge of the supervision and automation of the whole experiment. At the highest level the ECS "bubble" is decomposed in three entities (depicted in Fig. 14):

- The Run Control: Handles the Configuration, monitoring and operation of the complete Data Acquisition and Dataflow Systems.
- The Auto Pilot: When switched on automatically configures, starts and keeps the run going, detecting and recovering from most typical errors or problems.
- Big Brother: Is in charge of overall coordination. Depending on the LHC state it controls all sub-detector's voltages, opens and closes the VELO sub-detector and acts on the Run Control via the Auto Pilot.

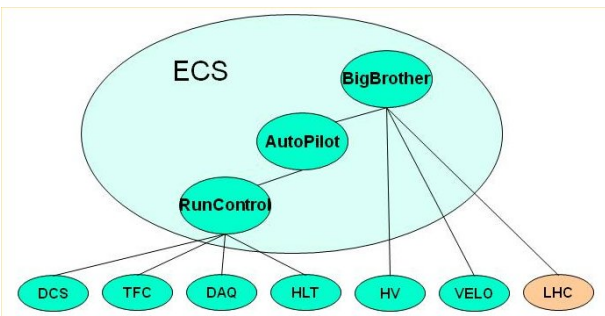

Figure 14: ECS top-level architecture.

With the introduction of the Split HLT paradigm, also the top-level control was enhanced. Since there are now three HLT slices this led to the introduction of three "independent" partitions each one with its own Run Control. Figures 15 and 16 show Big Brother and the three Run Control User Interfaces.

Each of the three Run controls can be operated independently but they each have their dedicated Auto Pilot and are normally automatically controlled by Big Brother.

The implementation of the three Run Control instances is identical; they inherit from a common class of rules and mechanisms. One important Run Control mechanism is described as the "Activity".

The Activity Concept

The Activity defines the set of parameters, we call it "recipe" that will be applied by all Sub-detectors and subsystems at Configure. The Activity name gets propagated down the control tree together with the Configure command and its usage follows a well-defined convention. For example an Activity can be called "PHYSICS|LEAD", when each sub-system receives the activity it will try to apply a recipe with the exact same name if one exists, if not it will remove the last part after and including the "|". In this example it would then try to apply "PHYSICS", if this one still does not exist it will then apply "DEFAULT". This allows the various sub-detectors and sub-systems to have more or less granularity in their settings and allows us at the top-level to easily create new derived Activities, which only affect the settings of small parts of the Experiment.

The Activity concept has been in place since the beginning of LHCb operation and includes also global high-level parameters, for example how many sub-farms are needed for a certain activity and which trigger configuration should be used. With the Split HLT also the Dataflow Architecture to be used became part of the Activity parameter set.

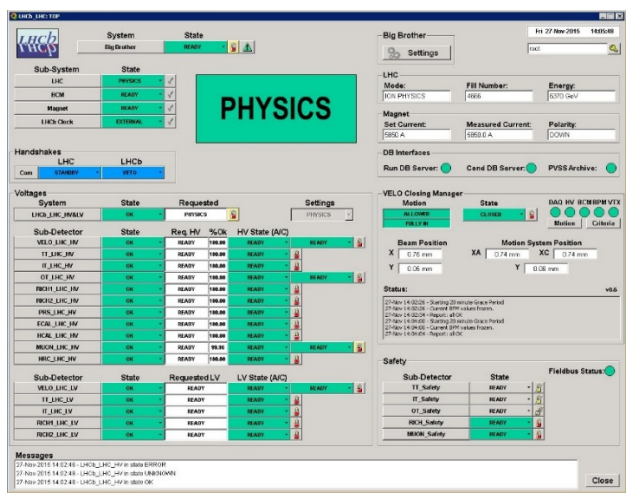

Figure 15: Big Brother User Interface.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THDPL01

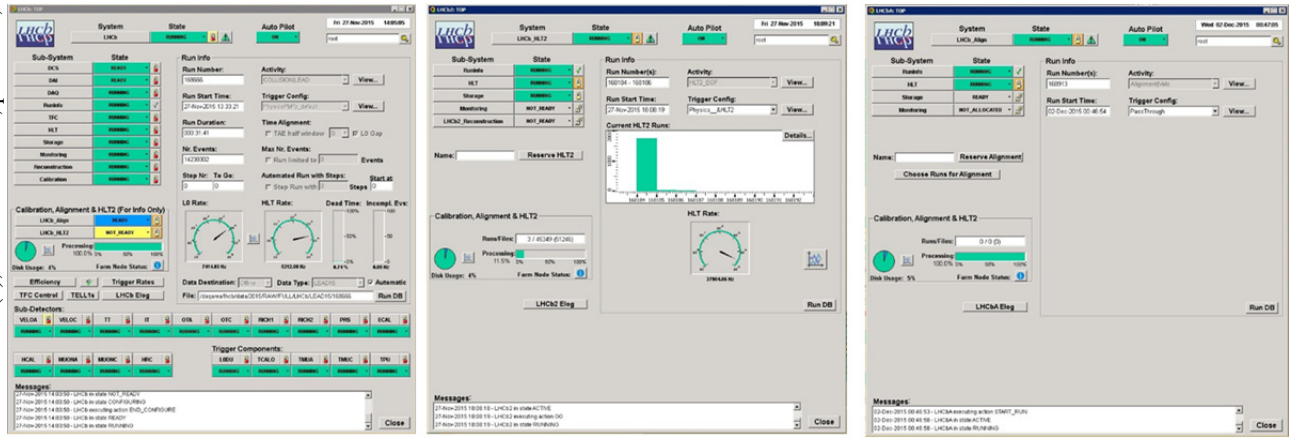

Figure 16: The main LHCb Run Control on the left, LHCb_HLT2 for HLT2 Run Control in the center and LHCb_Align for the Alignment control on the right.

Since Big Brother is the entity "in contact" with the LHC it is up to him to decide which Activity and/or which Scenario should be applied and when for each partition. In order to do this Big Brother uses what we call "Schedulers", configurable from his "Settings" button.

The Scheduler Concept

The scheduler defines the sequence of operations of a certain partition based on input events; these are normally the LHC state but can also be other pre-defined events. Possible operations on a partition can be starting/stopping an "Activity" or applying a "Scenario". Figure 17 shows the current scheduler definitions for the three partitions.

What is maybe worth noting is that with these few lines per partition, in particular 3 lines for the main Run Control, the complete LHCb operations are automated:

- LHCb starts a COLLISION Activity when the LHC is getting ready for delivering physics collisions. When collisions are over, it performs a short (10 minutes) "IVSCAN" activity required by the Silicon Sub-detectors and then goes into End-of-fill activity to keep the system running.
- HLT2 Runs automatically, permanently in the background, but changes number of tasks, load-balancing with HLT1 tasks, according to the LHC state.
- Several types of Alignment activities are performed in the whole farm in parallel with the previous activities, once pre fill, when enough events are collected of the respective required type. Each Alignment activity takes normally a few minutes.

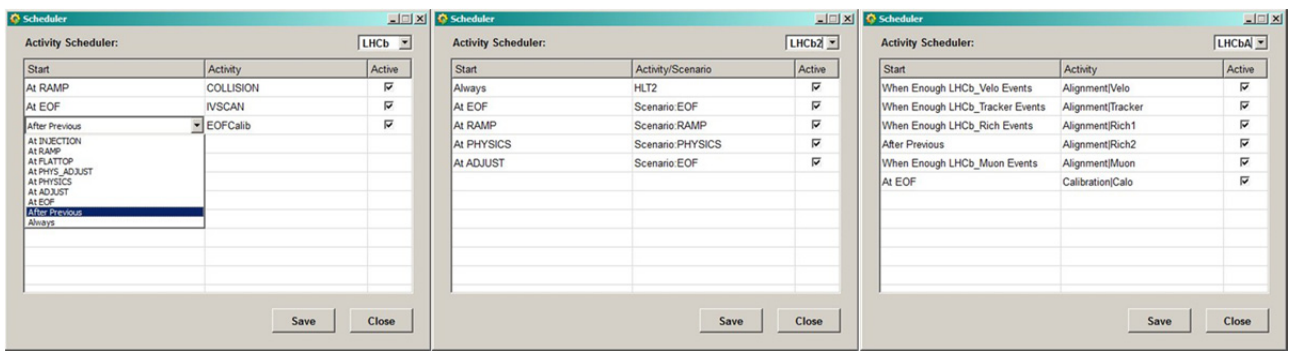

Figure 17: Scheduler Settings for the three partitions: LHCb on the left (with the list of possible input events), LHCb_HLT2 in the center and LHCb_Align on the right.

ISBN: 978-3-95450-193-9 doi:10.18429/JACoW-ICALEPCS2017-THDPL01

OFFLINE SIMULATION

In order to use the farm resources to the maximum, an extra activity can also be run in the HLT Farm: Offline Simulation. The LHCb offline computing environment is based on DIRAC (Distributed Infrastructure with Remote Agent Control)[7]. In DIRAC, job agents are started on Worker Nodes, they pull waiting tasks from the central Workload Management System, process them on the available resources and send the results back. An ECS control sub-system, using the same tools, was developed in order to bridge the offline and the online world by launching offline agents in the online farm nodes. Initially meant to be used only outside running periods when the farm was idle, it is now also used in parallel with the other activities.

Figure 18 shows the farm usage (in terms of HLT1, HLT2 and DIRAC jobs) during a period of around one month. Normally when the LHC is at nominal running conditions the farm is completely busy with HLT1 and HLT2 tasks, but due to a problem at the end of August the LHC couldn't run at its full capacity. At this point we started a few Simulation tasks (2 to 3 per farm node) in parallel with Physics running. Then in the middle of September there was a Machine Development and Technical stop period of around 10 days and there we first run HLT2 full blast to empty the local disks and then completely filled the farm with Simulation tasks, which we then stopped, leaving only a few per node, in view of the starting Physics campaign.

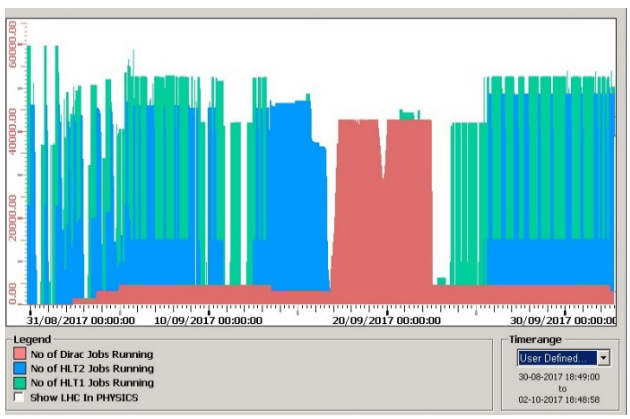

Figure 18: Farm Usage Graphic.

CONCLUSIONS

LHCb has implemented a homogeneous and integrated control system. Due to its modularity and flexibility, the integration of a completely different and quite complex dataflow pattern was not too difficult. Most mechanisms and concepts stayed the same but were extended to deal with the new parallel activities. Thanks to two new concepts: dataflow architectures and scenarios, allowing us to change operating mode within seconds, the farm can be used to the last CPU cycle. This allowed the Physicists to use the recovered CPU power for online Calibration and Alignment and to use its results to obtain a better trigger efficiency and better quality data directly out of the Online System.

The whole system is now completely automated allowing LHCb to continue being operated by a single operator.

REFERENCES

[1] LHCb Collaboration, "LHCb – the Large Hadron Collider beauty experiment, reoptimised detector design and performance", CERN, Geneva, Switzerland, Rep. CERN/LHCC 2003-030, 2003.

[2] D. R. Myers *et al.*, "The LHC experiments Joint COntrols Project, JCOP", in *Proc. Int. Conf. on Accelerator and Large Experimental Physics Control Systems (ICALEPCS'99)*, Trieste, Italy, Oct. 1999.

[3] S. Schmeling *et al.*, "Controls Framework for LHC experiments", in *Proc. 13th IEEE-NPSS Real Time Conference*, Montreal, Canada, 2003.

[4] PVSS-II/WinCC-OA, http://www.etm.at/index_e.asp

[5] B. Franek and C. Gaspar, "SMI++ - an object oriented Framework for designing distributed control systems", *IEEE Trans. Nucl. Sci.*, vol. 45, no. 4, pp. 1946-1950, 1998.

[6] G. Barrand *et al.*, "GAUDI : The software architecture and framework for building LHCb data processing applications", *Comput. Phys. Commun.*, vol. 140, pp. 45-55, 2001.

[7] A. Tsaregorodtsev *et al.*, "DIRAC: a community grid solution", *J. Phys. Conf. Ser.*, vol. 119, no. 6, p. 062048, 2008.