PROCEEDINGS OF SCIENCE

LHCb HLT2: Real-time alignment, calibration and software quality-assurance

Miroslav Saur[a,*]

[a] *On behalf of the LHCb collaboration*
Fakultät Physik, Technische Universität Dortmund,
Dortmund, Germany

E-mail: miroslav.saur@cern.ch

LHCb's second level trigger, deployed on a CPU server farm, not only selects events but performs an offline-quality alignment and calibration of the detector and uses this information to allow physics analysts to deploy essentially their full offline analysis level selections (including computing isolation, flavour tagging, etc) at the trigger level. This "real time analysis" concept has also allowed LHCb to fully unify its online and offline software codebases. We cover the design and performance of the system which will be deployed in Run 3, with particular attention to the software engineering aspects, particularly with respect to quality assurance and testing/limiting failure modes.

41st International Conference on High Energy physics - ICHEP2022
6-13 July, 2022
Bologna, Italy

*Speaker

© Copyright owned by the author(s) under the terms of the Creative Commons
Attribution-NonCommercial-NoDerivatives 4.0 International License (CC BY-NC-ND 4.0).

PoS(ICHEP2022)685

1. LHCb trigger design

The LHCb experiment during Run 3 is expected to operate at luminosity of 2×10^{33} cm^{-2}s^{-1} with on average 5 visible proton-proton collisions per bunch crossing [1]. To cope with this instantaneous luminosity the trigger system must be fast enough and at the same time highly flexible to cover the complex LHCb physics programme ranging from a core heavy flavour focus , electroweak and forward high p_{T} studies to heavy ion programme and more. This is achieved by employing a software-only trigger, split into two stages called High Level Trigger 1 (HLT1) and High Level Trigger 2 (HLT2), based on the real-time analysis approach with a full online reconstruction and selection. The trigger scheme and design is fully discussed at [2, 3].

Several conditions must be fulfilled to achieve the aforementioned goal. At first, a full alignment and calibration procedure, described at Sec. 2, running real-time is needed for achieving a high-purity selection in HLT2 and to ensure a consistent data processing chain. Then, in order to process all the data, HLT2 needs to be fast enough, which is accomplished by utilising several modern computing approaches as described in Sec. 3. Finally, to ensure stability of the system during development and data taking phase a robust quality assurance approach is deployed.

2. Alignment and calibration

Two procedures are distinguished at LHCb: alignment (VELO, RICH mirrors, UT, SciFi, Muon) and calibration (RICH, ECAL, HCAL[1]). Both operations are triggered at the beginning of each LHC fill and the procedure is fully implemented in the LHCb control system. Dedicated HLT1 trigger lines are used to select events interesting for the alignment and calibration procedure. Selected HLT1 events are then stored in a buffer until the necessary statistics is accumulated for the alignment and calibration procedure, which runs before the HLT2 is executed. Using this strategy any HLT2 reconstruction can then use the most relevant and precise alignment and calibration constants. For example, in Run 2 typical VELO alignment samples consisted of 50 000 minimum bias events.

Tracking detectors (VELO, UT, SciFi, Muon) are aligned by minimising the χ^2 of all tracks with respect to the chosen alignment parameter α which accounts for translations and rotations of the detector elements. This minimisation procedure is using the iterative Newton-Raphson method by calculating the first and second derivatives of χ^2 with respect to the α [4]. Tracks used for the alignment procedure are obtained from a Kalman filter as used in the track reconstruction with an additional possibility to add a vertex and mass constraints to increase precision [5].

Technically, the alignment procedure is based on two parts: *Analyzer* and *Iterator*. The Analyzer runs the track reconstruction, computes the χ^2 derivatives and saves them to binary files. The Iterator then collects the derivatives, performs the minimisation step and does a convergence check. If a significant difference is found between previous and new alignment constants, the updated constants are used in HLT2. As the Analyzer is using multi-threaded reconstruction based on 163 nodes it is expected that the time required to perform alignment should be similar as in Run 2.

[1]Calibration of the HCAL is not done in real-time.

PoS(ICHEP2022)685

3. Real-time reconstruction in HTL2

Original LHCb event model for Run 1 and 2, meaning all classes representing the full data flow from the DAQ to the output files used for a further data analysis, was based on the general Object-Oriented (OO) computing scheme. All objects in the original OO scheme were a "keyed containers", i.e. each object was represented by a container and the relevant key. Keyed containers were implemented as an array-of-structures (AoS), allowing a simple usage. However, old event model was susceptible to many small memory allocations, random jumps when accessing memory and also objects were often copied and moved, thus requiring an additional memory and slowing down the data processing. In order to fully use the advantage of modern fully software trigger, the event model for Run 3 had to be completely rewritten with a focus on speed and flexibility. For those reasons the new LHCb event model is based on the Struct-of-Arrays (SoA) layout allowing to read only slices of data that are needed and is natively compatible with Single Instruction Multiple Data (SIMD) instructions set [3, 6].

Using SIMD allows to fully utilise vectorisation which is crucial to achieve the required HLT2 throughput. One of the examples can be found in the *forward tracking*, which is one of the two long[2] tracks reconstruction algorithms at LHCb. The Forward Tracking is searching for a forward extension in SciFi for any given input VELO track. Based on the SciFi geometry, the forward extension is a set of hits form a different SciFi layer, in total 10 to 12 hits. The algorithm searches for a slightly curved trajectory, due to a low-level magnetic field within the SciFi volume. A Hough-like transformation is applied to identify a proper pattern of SciFi hits matching a VELO track [7]. This is a highly computing-intensive task where parallelization can be successfully applied in searching for the best combinations of SciFi hits and VELO track, as several combinations can be scanned at the same time. Using the Advanced Vector Extensions 2 the Forward Tracking throughput is improved by 60% without losses in reconstruction efficiency or increasing fake track fraction [7].

During Run 2 operations selection and reconstruction took roughly 70% and 30% of the HLT2 stage processing time, respectively. The same time division is expected for Run 3 with currently implemented $O(1500)$ HLT2 selection lines. The core part of the new SoA-based selection framework are new Throughput Oriented (ThOr) functors, function objects designed to be agnostic to input and output type as well as object type which they are applied on. A significant advantage of ThOr functors is their composability which allows to create a new functionality by a simple chaining of basic functors, for examples the x coordinate of the particle decay vertex can be obtained as: $Particle.vertex().position().x$. Although this system was developed to work both with the old and new selection vertex, usage with the SoA-based event model achieves the highest gain.

Additional speed up is obtained when a functor cache is used instead of re-compiling functors. The ThOr functors are written in C++, but the HLT2 selection lines, and consequently options for ThOr functors, are written in Python. To run selections within HLT2 the analyst-defined selection is then translated and compiled into executable C++ code. Functors then can be used within HLT2 without need for any further interpretation. This is done by using GCC just-in-time compiler. Difference in compilation time between using the old implementation and using functor cache in the different scenarios is shown in Table 1.

[2]Long track is defined as a track with hits in VELO, UT (optionally) and SciFi.

PoS(ICHEP2022)685

Compile	Basic compiler (s)	GCC JIT and cache (s)
Single functor	11	13
5 different functors	20	13
Typical HLT2 selection	70	24
Compilation rerun	70	0

Table 1: Comparison of compilation time using the old implementation and GCC just-in-time (JIT) compiler together with the cache.

Taking Bandwidth [GB/s] ≈ signal rate [kHz] × event size [kB], with a hardware constraints on the output bandwidth and expected signal event rate, the event size is the only free parameter which can be optimised in order to record a higher statistics. One of the possibilities to reduce the event size is the Turbo model extensively used by LHCb during Run 2 period [10]. In this model only the information needed for a physics analysis is stored and remaining data are discarded. Turbo model evolved significantly during Run 2, where original Turbo allowed to store objects associated to individual reconstructed decays, corresponding to the event size of O(10 kB). Later development allowed to store a fully reconstructed event while still removing all the raw information, with event size of O(100 kB). Finally, the last iteration during Run 2 allowed to specify which exact objects are to be stored for later analysis, resulting into event size in range O(10 - 100 kB), but generally still lower than the corresponding event to be fully saved. Where in Run 2 Turbo corresponded to roughly 30% of the trigger rate, for Run 3 Turbo is the baseline approach responsible for roughly 70% of the trigger rate.

Breakdown of the full HLT2 event throughput, without a physics selection, is shown in Fig. 1. This throughput was obtained using an optimised sequence with removed redundancy in the Long tracks reconstruction, using a partially parametrised Kalman filter for material scattering and with improved matching between tracks and ECAL clusters [11].

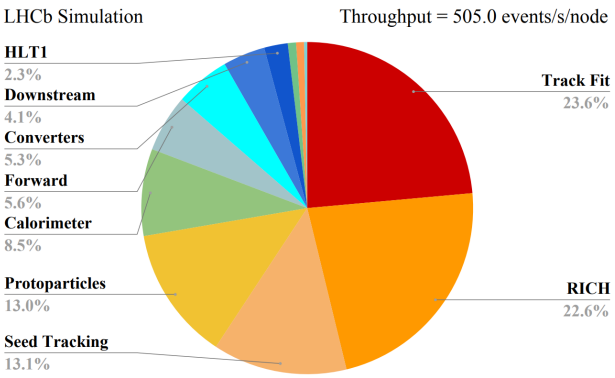

Figure 1: Breakdown of the event throughput of the HLT2 reconstruction for LHCb in Run 3, using an optimised configuration. Reproduced from [11].

PoS(ICHEP2022)685

4. Software quality-assurance

The LHCb software stack is a highly modular system based on the GAUDI framework with code being hosted on GitLab with a policy where each member of the LHCb collaboration can contribute to the code development. This leads to a concurrent code development where any new contribution is represented by a Merge Request (MR) on GitLab. To ensure functionality of any new contribution and the code quality, a dedicated working package focusing on the maintenance and code quality assurance was created and is an integral part of the development cycle.

The review policy before merging any new MR consists of a full review of the code and subsequent test. Code review depends on the code maintainers, which are typically senior LHCb software experts responsible for the full code review, and shifters, junior members. Shifters help with checking requirements of each MR, triggering and evaluating continuous-integration tests. Description of each new contribution should be written as accessible to shifters including the code documentation, that way shifters can learn more about the LHCb codebase, which also serves as an internal software training within the collaboration.

Testing infrastructure at LHCb consists of two related parts: The LHCb nightly build system, which is based on $O(300)$ cores and builds the full software stack every night to check if code compiles, ran internal tests successfully and finalises without errors. This test can be triggered for any MR directly from GitLab using web-hooks [12].

Second part of the testing infrastructure is the LHCb Performance Regression (LHCbPR) [13], which uses the same infrastructure as the nightly system. In the case of the software QA, LHCbPR is used for running dedicated HLT1 and HLT2 reconstruction tasks over a simulated Monte Carlo samples. Typical output of these tasks are reconstructed values such as track and vertex properties, kinematics of reconstructed tracks and similar. Results are then visualised using the web-based LHCbPR front-end which allows a quick check and comparison of resulting values [14].

Important part of any code development is also internal software training. During the new trigger development, around 25 dedicated upgrade software session hackathons were held in last 6 years, focusing both on the new trigger framework development and training new contributors in all relevant technical aspects such as general modern computing methods, programming of heterogeneous platforms and many others. Advanced computing skills are already necessary for any modern particle physics experiment and a community-wide effort is needed to keep and spread necessary knowledge and also keep computing experts within the field.

5. Conclusion

A new fully software trigger was developed for the LHCb upgrade as the nominal strategy for Run 3. HLT2 part of the trigger is built on modern computing methods such as using Struct-of-Arrays based event model, vectorisation and multi-threading. Turbo model has been chosen as the baseline for Run 3 in order to record a higher statistics given a hard bandwidth limits. Extended code quality assurance and testing system is well established and integrated into the development cycle. HLT2 is working and ready for Run 3 data taking.

PoS(ICHEP2022)685

Acknowledgements

The author would like to acknowledge support by the LHCb collaboration and in particular thank the RTA, Simulation and LHCb Computing teams.

References

[1] LHCb collaboration, *Framework TDR for the LHCb Upgrade: Technical Design Report*, CERN-LHCC-2012-007, LHCb-TDR-12, https://cds.cern.ch/record/1443882

[2] LHCb collaboration, *LHCb Trigger and Online Upgrade Technical Design Report*, CERN-LHCC-2014-016, LHCB-TDR-016, https://cds.cern.ch/record/1701361

[3] LHCb collaboration, *Computing Model of the Upgrade LHCb experiment*, CERN-LHCC-2018-014, LHCB-TDR-018, https://cds.cern.ch/record/2319756

[4] F. Reiss, *Real-time alignment procedure at the LHCb experiment for Run 3* , LHCb-PROC-2023-001, http://cds.cern.ch/record/2846414

[5] J. Amoraal et.al., *Application of vertex and mass constraints in track-based alignment*, Nucl. Instrum. Meth. A 712 (2013), https://doi.org/10.1016/j.nima.2012.11.192

[6] S. Esen, A. M. Hennequin, M. De Cian, *Fast and flexible data structures for the LHCb Run 3 software trigger*, LHCb-PROC-2022-012, https://cds.cern.ch/record/2824161

[7] P. A. Gunther, *LHCb's Forward Tracking algorithm for the Run 3 CPU-based online track reconstruction sequence*, LHCb-PROC-2022-009, https://cds.cern.ch/record/2819858 arXiv:2207.12965

[8] LHCb collaboration, *Comparison of particle selection algorithms for the LHCb Upgrade*, LHCB-FIGURE-2020-018, https://cds.cern.ch/record/2746789

[9] E. Govorkova et al., *A new scheduling algorithm for the LHCb upgrade trigger application*, J. Phys.: Conf. Ser. 1525 012052, 10.1088/1742-6596/1525/1/012052

[10] R. Aaij et al., *A comprehensive real-time analysis model at the LHCb experiment*, JINST 14 P04006, https://iopscience.iop.org/article/10.1088/1748-0221/14/04/P04006

[11] LHCb collaboration, *HLT2 reconstruction throughput and Forward Tracking performance for Run 3 of LHCb*, LHCB-FIGURE-2022-005, https://cds.cern.ch/record/2810226

[12] R. Curie, R. Matev, M. Clemencic, *Evolution of the LHCb Continuous Integration system* ,EPJ Web Conf. (2020) 245, https://cds.cern.ch/record/2752846

[13] D. Popov, *Testing and verification of the LHCb Simulatio*, EPJ Web Conf. (2019) 214, https://cds.cern.ch/record/2728528

[14] Y. Hou et al., *Monitoring reconstruction software in LHCb*, EPJ Web Conf. (2021) 251, https://doi.org/10.1051/epjconf/202125103044

PoS(ICHEP2022)685