The LHCb software and computing upgrade for Run 3: opportunities and challenges

C Bozzi\textsuperscript{1,2} and S Roiser\textsuperscript{1},
on behalf of the LHCb Collaboration

\textsuperscript{1} European Organization for Nuclear Research (CERN), Geneva, Switzerland
\textsuperscript{2} Sezione INFN di Ferrara, Ferrara, Italy
E-mail: concezio.bozzi@cern.ch

Abstract. The LHCb detector will be upgraded for the LHC Run 3 and will be readout at 30 MHz, corresponding to the full inelastic collision rate, with major implications on the full software trigger and offline computing. If the current computing model and software framework are kept, the data storage capacity and computing power required to process data at this rate, and to generate and reconstruct equivalent samples of simulated events, will exceed the current capacity by at least one order of magnitude. A redesign of the software framework, including scheduling, the event model, the detector description and the conditions database, is needed to fully exploit the computing power of multi-, many-core architectures, and coprocessors. Data processing and the analysis model will also change towards an early streaming of different data types, in order to limit storage resources, with further implications for the data analysis workflows. Fast simulation options will allow to obtain a reasonable parameterization of the detector response in considerably less computing time. Finally, the upgrade of LHCb will be a good opportunity to review and implement changes in the domains of software design, test and review, and analysis workflow and preservation. In this contribution, activities and recent results in all the above areas are presented.

1. Introduction
The LHCb experiment will be upgraded for data taking in Run 3 and after \cite{1}. The instantaneous luminosity will increase by a factor five, from $4 \times 10^{32}$ to $2 \times 10^{33} cm^{-2}s^{-1}$. In order to fully profit from the increased luminosity, the current L0 hardware trigger will be removed, and a full software trigger will be deployed, with the goal of sustaining trigger capabilities up to the inelastic event rate of 30 MHz. The full read-out of the detector at this rate has a major impact on software and computing systems.

A study of the trigger output rates for different physics scenarios is reported in the LHCb Trigger and Online Upgrade TDR \cite{2}. In summary, output bandwidths between 2 and 5 GB/s are expected, to be compared with 400 MB/s in Run 1 and 700 MB/s in Run 2. If the current computing model and software framework are kept, the data storage capacity and computing power required to process data at this rate, and to generate and reconstruct equivalent samples of simulated events, will exceed the current capacity by at least an order of magnitude. This challenges several areas of the LHCb software and computing infrastructure, that will not scale as needed. Moreover, the computing resources which are likely to be funded in the upgrade area
will not be sufficient to accommodate the requirements. It is therefore mandatory to study how the current model can be changed into a sustainable one.

The introduction of the split High Level Trigger (HLT) concept in the Run 2 data taking [3] has allowed the integration of real-time calibration and alignment into the data taking process [4]. The online reconstruction is therefore equivalent to the offline one, thus enabling analyses to be performed on physics objects produced directly out of the trigger. Furthermore, the introduction of the Turbo data stream, where only a subset of the event information is saved, allows to decrease the event size, and therefore the output bandwidth and the need for additional offline computing resources. The physics program at LHCb will be clearly maximized by further exploiting these concepts, which ultimately means to move the event reconstruction and selection as close as possible to the online farm, and to implement data streaming as early as possible in the computing model. This implies that the event filter farm should be capable of performing such tasks very efficiently and very rapidly.

A major challenge to be faced in the upgrade era is therefore the efficient usage of computing resources. Multi-core and many-core architectures, as well as coprocessors such as GPGPUs and FPGAs, offer significant speedups, making them particularly suited for the event filter farm. The efficient use of these new processors requires a paradigm shift in the LHCb core software framework, scheduling, and event model. The R&D activities related to this topic are presented in Section 2.

The events accepted by the trigger will be distributed for analysis in a similar way as today through the Worldwide LHC Computing Grid (WLCG). Section 4 details the challenges to be dealt with in this domain. Given that event reconstruction and selection will happen at the trigger level, the event format will be compact, and comparable to the ones currently used (Data Summary Tape, DST or Micro Data Summary Tape, µDST). Offline data processing will be very limited, and the storage costs for the recorded data will be driven essentially by the HLT output rate. The vast majority of offline CPU work will be therefore dedicated to the production of simulated events. As the number of simulated events is generally proportional to the number of data events, it follows that the work needed for simulation will exceed the expected resource increase considerably. It is therefore necessary to speedup simulation and pursue alternative ways, where faster or parameterised simulations are employed for quick analysis prototyping, background studies, and the estimation of (part of) systematic uncertainties, while full simulation is granted to the study of signal efficiencies only. Section 3 summarizes the work plan for these activities.

Section 5 mentions the challenges that need to be addressed in order to strengthen collaborative working. A summary is given in Section 6.

2. Core software framework

The stringent requirements on the data processing throughput of the LHCb upgrade imply that the utilization of computing capacities of current and emerging hardware has to be improved both quantitatively and qualitatively. The latter must encompass functional improvements, such as engaging previously unused dimensions of computing resources (threads, co-processors), but also non-functional ones (ensuring the scalability of the framework and its applications).

2.1. Framework

The data processing model of the Gaudi framework [5, 6] used in LHCb implies that events are processed sequentially, as well as that the sequence hierarchies, and all Gaudi algorithms within each sequence, are executed sequentially for each event. With such data processing model, only the multiprocessing approach is possible, with consequent fundamental limitations on throughput and scalability, such as weak scalability in RAM usage, inefficient handling of CPU-blocking operations, e.g. disk/network I/O and offload computations. To mitigate these
limitations, several techniques and models are considered, such as simultaneous multithreading, a task-based programming model [7], by which the basic computation units, e.g., the Gaudi algorithms, are represented and handled as tasks, and concurrent data processing, with inter- and intra-event, and (optionally) intra-task concurrency. The sustainability and conformity of the above-mentioned principles in the Gaudi framework were demonstrated by a prototype [8], which constitutes the building block of the framework for the upgrade.

This new paradigm requires the development of new Gaudi components, such as the Gaudi task scheduler [9], that is central to the operability and viability of the approach [10]. Implementing a framework prototype, as well as the upgrade of all associated data processing applications, at production level requires a deep revision of the corresponding code base. In particular, a refactoring of many existing components is necessary to make them compliant with the requirements of thread safety and reentrancy. This is compulsory to ensure the general functional correctness of the framework and of its applications in a concurrent environment. This is also an important integral part of the measures that need to be taken in order to achieve good scalability.

Another wide-ranging change will concern the mechanisms of declaration of data dependencies between the Gaudi Algorithms. In particular, their input and output data requirements must be explicitly declared to the framework. This is needed by the Gaudi task scheduler for task concurrency control.

The concurrent paradigm places several stringent constraints on the event data model. First and foremost, once data is made available and written to the Transient Event Store (TES), it must remain immutable. This would allow to avoid problems that are difficult to debug and would significantly simplify the process of concurrency control by allowing to focus on resolution of control and data flow rules.

A functional framework was developed with the above requirements in mind, including a read-only TES, in the context of Gaudi. About one hundred algorithms have been adapted, some of them were partially ported to be used in the current framework, with significant gain in performance. An application was built (Mini-Brunel), that includes a Kalman filter for charged track reconstruction. Preliminary measurements of the performance show a good scalability in terms of multi-threading, a very low and scalable memory footprint, with a baseline short of 0.9GB and an additional 30MB per core, and a speed-up factor that scales almost linearly with the number of cores, with only 12% loss of performance with respect to the theoretical maximum when using all physical cores, gradually increasing to 20% when using 2x-hyperthreading on an eight-core machine, and to 30% with a 2x-hyperthreading on a many-core machine. Further measurements and monitoring of time spent on single algorithms and tuning of the scheduler will allow to identify hotspots and strategies to optimize the LHCb software.

2.2. Event Model
The LHCb event data model [11] has been very successful in enabling developers to efficiently write reconstruction or selection code, by providing e.g. an object representation of raw data, reconstructed objects and decay trees, and limiting the number of changes in the interface to the data model. However, as a result of the Array of Structures (AoS) design of the event model classes in its current form, the exploitation of SIMD instruction is more difficult. In addition, current event model objects are not composable, so if information needs to be added to objects without modifying them, they need to be copied. This costs significant amounts of memory, which in turn leads to sub-optimal usage of resources.

As a result of the changes to the core framework, the event model objects are therefore required to become read-only after their initial creation, to be composable, to allow the choice of memory layout (SoA, AoS, etc.), to use single precision where possible.

At the time of this writing, the TES was already made read-only, mainly by adapting code
by splitting objects. Composition was implemented by using the range v3 library [12], which provides enough functionality for transient data, and has been successfully used to port the reconstruction of photon in the LHCb Cherenkov detectors. The implementation of generic composition, a major complex task that would not be easily manageable for the end users, might not be needed at all. For instance, object composition in non-transient data could be handled in the packing step, that transforms event model objects between their in-memory and on-disk representations in order to efficiently store them.

A preliminary test of switching from double to floats in a vectorized version of the Kalman filter has also been performed, with a factor two improvement in speed [13]. However, a thorough validation of the physics performance and the benchmarking of memory layouts at the algorithm level are mandatory and need to be addressed.

2.3. Detector Description and Conditions Database

The LHCb Detector Description (LHCbDD [14]) is based on a home-made framework, developed along the lines of the Geant 4 geometry, with special extensions to implement generic active volumes. The persistent format chosen was XML. The current implementation has several limitations, in particular it is not thread-safe, and therefore has negative impact on the throughput performance of a multithreaded software framework.

The DD4Hep toolkit [15] has been investigated as a replacement. It has been shown[16] that using DD4Hep from Gaudi is relatively easy. Indeed, the full LHCb geometry has been automatically converted and tested, the next step being validation in terms of visualization, computation of material budget, mapping a space point to a volume, local-to-global transformations, usability in alignment procedures. In addition, a geometry where the detailed structure of the LHCb detector is averaged out over a small number of elements, has been defined and is available. Its usage in algorithms that heavily rely on particle propagation, such as a Kalman filter, will dramatically increase the throughput.

The LHCb conditions data are currently stored in a database managed with the Cool/Coral libraries [17], developed by CERN/IT in collaboration with LHCb and ATLAS. The level of thread-safety of the Cool/Coral library is unclear, but it is unlikely that two concurrent threads can query the CondDB. Moreover, the XML format used for persistency is slow to read and not compact. The transient representation of conditions data are also likely to change, in order to speed up code performance.

In order to adapt the conditions interfaces to a multithreaded environment, a proposal and a prototype have been developed in the context of Gaudi, and will be discussed and validated in that context. A prototype to manage the conditions backend with git is also available, that offers better performance and easier maintainance than other storage database solutions, and allows to drop a significant amount of code. The usage of git as a management tool for conditions should be carefully evaluated in the context of the online environment.

2.4. Optimization

The task-based framework described above has been successfully used on multi- and many-cores architectures. Further studies have been performed on alternative architectures such as GPGPUs and FPGAs, with promising results in terms of throughput [18, 19]. However, a large investment in code rewrite should be taken into account and a cost-benefit analysis should be performed before taking decisions. Also, the reproducibility of results obtained on these alternative architectures with more conventional ones should not be overlooked, since e.g. simulation will run mainly on the latter.

Performance improvements are expected by taking advantage of wide processing units and improved scheduling in Gaudi, and by minimizing cache misses. Examples of parallelism in the
LHCb reconstruction software are given in [13, 20]. Monitoring and in-depth measurements in these domains are challenging but nevertheless needed in order to make substantial progress.

3. Simulation

In the current offline processing, the majority of CPU work (about 70%) is spent for Monte Carlo simulation. The current simulated samples correspond to about 15% of the total data statistics. Given that in the LHCb upgrade the trigger purity will increase, it is expected that a larger fraction of events will have to be simulated. This, together with the increased luminosity and the more complex nature of the events, results in an increase between one and two orders of magnitudes of needed CPU work, that cannot be sustained in absence of a corresponding increase of computing resources in the next future.

Two avenues are being pursued in order to mitigate the required computing resources: the usage of fast simulations, where speed-up is obtained with either fully parameterized, or fast detector response, or reuse of events, and the usage of parallelised simulation frameworks (multi-threading, multi-processor) and of geometries of different complexity.

The flexibility of the LHCb Simulation framework, Gauss [21], allows to implement, in addition to the full Geant4 simulation [22, 23], a variety of predefined safe and easy to use alternative configurations. The most appropriate configuration can then be applied to a given analysis according to e.g. lower accuracy for specific particle types in exchange for faster processing time. Many options are possible including simulating only part of an event, replacing a full Geant4 simulation with faster versions for a given detector (fully parametric, based on a database library or faster Geant4 implementations) or disabling specific processes (e.g. Cherenkov effect in RICH detectors), stopping particles at a specific stage, reducing the geometry (e.g. removing sub-detectors), re-using the underlying event or merging a simulated signal with real data, or even using a fully-parametric simulations providing reconstructed objects and using as much as possible available packages like Delphes [24]. Some of these options are already present in Gauss, allowing to explore their impact on the physics analysis.

In addition to providing alternative faster solutions, the simulation software itself needs to be made faster, and an in-depth code optimization should be carried out. Therefore, it is essential to make detailed timing measurements of Gauss and to review some of the simulation settings, where more fine grained solutions may be appropriate: an evaluation of the optimal cuts for each detector region and particle type should be carried out to obtain an improved computing performance with the same or better physics modelling.

Another important aspect is enabling concurrency in the simulation application. This would allow the usage of multi- and many-core architectures. The main player here is Gaudi with its multi-threaded version, see above, and its proper coexistence with the multi-threaded version of external tools like Geant4. Integration tests have been done by ATLAS [25], where some design choices have been made to make the different concurrent models of Gaudi and Geant4 work together. LHCb will start by adopting similar choices. Nevertheless all the LHCb simulation specific code will need to be adapted. This and further evolutions will need to be fully tested. The effective improvement given by the parallelization will strongly depend on the implementation details of multi-threading and multi-processing in a distributed environment.

In any simulation with complicated geometry, the majority of time is spent navigating the geometry itself. New geometry packages with improved performances have become available for Geant4, like USolid [26]. This alternative version of Geant4 geometry should be tested in the Gauss framework. In addition, the outcome of the evaluation on the change in the detector description (see Section 2) would have an impact on how Gauss transfers the geometry information to Geant4. In any case simplified geometry descriptions, either via utilities provided by Geant4 or the LHCb detector description should be evaluated in the context of their use for some of the fast simulation options. A fully vectorized geometry package, VecGeom, is being
developed in the context of GeantV [27], the vectorized version of Geant4. While a first version of GeantV itself it is not expected to become available before 2018, VecGeom will be deployed in earlier versions of Geant4. These newer options should be tested as soon as they will become available.

On a broader view, LHCb started a collaboration with the software developers of the Future Circular Collider (FCC) project to create Gaussino, an experiment-independent version of Gauss, with the intention that Gauss will be based on it and provide the LHCb specific functionality. The development of Gaussino should allow an easier way to test new design and package options, like GeantV, with simpler settings but in an environment very similar to that of LHCb. A first version of Gauss based on Gaussino will also allow to explore more efficiently the optimal way to parallelize the simulation.

4. Distributed computing and data analysis

In the data flow model used in Run 1, the full raw event information is kept up until the end in the processing steps, the stripping, in which a selected subset of triggered events is provided to users for physics analysis. This model implies a heavy use of offline CPU, for reconstruction and stripping campaigns, and storage resources for raw data). Moreover it requires data to be moved from tape to disk and vice-versa at each stripping or reconstruction campaign to allow processing. One important advantage of this model is that the full event information is always saved (either on disk or on tape), thereby allowing for improvements in the reconstruction and selection criteria to be introduced at any moment.

Although very robust and well oiled, the Run 1 data flow model cannot be sustained in the upgrade era. The projected computing resources will not allow for the storage of the raw event information for the entire collected luminosity and the processing time of reconstruction and stripping will be comparable to the time required to collect the data. Therefore the whole data processing, from the data acquisition to the final physics objects, needs to be changed.

The concepts of split HLT and Turbo stream, introduced in the Run 2 data taking and already described in Section 1, will be further exploited in Run 3, where the fully software trigger system will give a dramatic increase in efficiency for most physics channels and the current stripping step will effectively happen online, thereby allowing for quick analysis turnaround time and resource optimization.

The building blocks for distributed computing and analysis in the upgrade era are detailed below. All of them can be already investigated with the current software framework.

4.1. Turbo stream to become the default

The capabilities of the Turbo stream should be extended in order to produce different output types, as discussed in Section 4.2. The benefit of this approach is essentially the removal of unnecessary information earlier in the workflow, with a clear benefit for online and offline resources optimisation.

4.2. New event formats

Currently, the output of the stripping can be either in the µDST or the DST formats. For both of them, parts of the raw event can be selectively saved as needed by analysts. In the past years, µDST has become the most widely used format (90% of the events available to physics analysis). In the µDST only the selected decay candidate and related information (such as PV and multiplicities) are saved in the output. In addition, other quantities can be calculated during the stripping job and saved in µDST but no additional information can be extracted after stripping. On the other hand, in the DST all tracks and calorimeter clusters are saved, therefore this mode is more suitable to study partially reconstructed decays, spectroscopy or for analyses that require full event information in general. On average, the µDST occupies ten times less disk...
space than a DST. It is known that, although several analyses cannot be performed on \(\mu\)DST, full event information is still not needed in most cases. Therefore it is necessary to investigate new data formats in between \(\mu\)DST and DST where, for example, only a cone of tracks around the selected candidate is stored such that storage usage is optimised without compromising the physics analysis.

4.3. Centralised ntuple production and alternative approaches
It is foreseen that ROOT ntuples will still be heavily used to perform data analysis. Currently, ntuples are produced by users in an unscheduled activity by running over Turbo or stripping output that is clearly not used directly for analysis. The possibility to have a scheduled, centralised ntuple production organised in trains, with each ntuple/analysis being a wagon, is being investigated, based on the work done within ALICE [28]. These productions would be centrally managed in a timescale of weeks. It should also be noted that (\(\mu\))DSTs can also be directly used for physics analysis, but this requires at the moment the entire LHCb software stack, thus making their usage unpractical. The development of a light-weight library and extensive documentation would greatly improve the situation.

4.4. Distributed computing
Infrastructures such as Dirac [29] will still be the main tool through which data is processed and distributed to the users. Given that the modular architecture of Dirac allows for adiabatic improvements of components, no major changes are expected in this area but rather an evolution from the current status in order to cope with the upgrade conditions. In parallel, more dynamic and flexible ways of data placement will be investigated, for example by monitoring the accesses to a given dataset.

5. Collaborative Working
Analysis in the LHCb upgrade era will be more complex, with larger data volumes to analyze; moreover, the techniques to write efficient and scalable software for new processor architectures will be more sophisticated than what is currently required. In order to avoid increasing the gap between software experts and analysts and enable any member of the collaboration to contribute to software development and profit from any new software tool or analysis methods, collaborative tools and working practices should be put in place.

A set of topics of interest for LHCb has been identified, which can be divided into two groups: those involving analysis (reproducibility and documentation) and those specifically related to software (design, review, testing and contributing). The former includes automatic analysis workflows, code reusability, and analysis documentation, preservation and replication via the CERN Analysis Preservation (CAP) framework [30]. On the software side, LHCb successfully made the transition to git and gitlab [31], that allow easier parallel developments and the introduction of code review procedures. The definition of a set of guidelines for software testing and code reviewing, and the creation of a handbook on how to contribute to LHCb software will increase the number of people contributing to LHCb software development.

To systematize profiling of LHCb applications and to help developers to evaluate how their changes behave in default test cases, a new service, LHCb Performance and Regression [32] has been developed, is already in production and tutorials have been organized to promote it within the collaboration.

6. Conclusions
The LHCb experiment will be upgraded for the Run3 data taking in 2021 onwards. A review of the ongoing R&D work in the domains of software and computing for the upgrade has been given in this paper.
The performance of the software trigger in terms of data throughput will be optimized by changing significantly basic building blocks such as the core framework, that will be task-based, the event model, that will be adapted to efficiently use wide processing units, and non-event data. Such changes will also be useful to improve the simulation framework. In other areas, such as the distributed computing, the analysis model, the implementation of alternative, faster simulations, and collaborative working, there will be a natural evolution towards the upgrade era, with current Run 2 data taking to be used as testbed.

Acknowledgments
The authors would like to acknowledge the contributions of R. Aaij, S. Amerio, M. Cattaneo, R. Cenci, P. Charpentier, M. Clemencic, A. Contu, G. Corti, B. Couturier, C. Haen, S. Neubert, G. Raven, I. Shapoval, F. Stagni in the development of the work presented in this paper.

References
[1] Aaij R et al. (LHCb collaboration) 2012 Framework TDR for the LHCb Upgrade: Technical Design Report LHCb-TDR-012
[2] Aaij R et al. (LHCb collaboration) 2014 LHCb Trigger and Online Upgrade Technical Design Report LHCb-TDR-016
[3] Raven G 2016 J. Phys. Conf. Ser. These proceedings
[4] Martellini M 2016 J. Phys. Conf. Ser. These proceedings
[5] Mato P 1998 GAUDI-Architecture design document Tech. Rep. LHCb-98-064 CERN Geneva URL https://cds.cern.ch/record/691746
[6] Barrand G et al. 2001 Comput. Phys. Commun. 140 45–55
[7] Task-based Programming https://software.intel.com/en-us/node/506100
[8] 2012 The Concurrent Framework Project (CF4Hep) http://concurrency.web.cern.ch/GaudiHive
[9] Shapoval I et al. 2015 IEEE Nuclear Science Symposium & Medical Imagine Conference Record (IEEE)
[10] Shapoval I 2016 Adaptive Scheduling Applied to Non-Deterministic Networks of Heterogeneous Tasks for Peak Throughput in Concurrent Gaudi Ph.D. thesis UNIFE URL https://cds.cern.ch/record/2149420
[11] Roiser S 2003 Event data modelling for the LHCb experiment at CERN Ph.D. thesis Vienna, Tech. U. URL https://cds.cern.ch/record/692288
[12] https://github.com/ericniebler/range-v3
[13] Campora D 2016 J. Phys. Conf. Ser. These proceedings
[14] Ponce S, Mato Villa P, Valassi A and Belyaev I 2003 eConf C0303241 THJT007 (Preprint physics/0306089)
[15] https://github.com/KIDASoft/DD4hep
[16] Clemencic M and Karachaliou A 2015 J. Phys. Conf. Ser. 664 072012
[17] Valassi A, Basset R, Clemencic M, Pucciani G, Schmidt S A and Wach M 2008 Proceedings, 2008 IEEE Nuclear Science Symposium, Medical Imaging Conference and 16th International Workshop on Room-Temperature Semiconductor X-Ray and Gamma-Ray Detectors pp 3021–3028
[18] Faerber C 2016 J. Phys. Conf. Ser. These proceedings
[19] Gallorini S 2016 J. Phys. Conf. Ser. These proceedings
[20] Stahl M 2016 J. Phys. Conf. Ser. These proceedings
[21] Clemencic M et al. 2011 J. Phys. Conf. Ser. 331 032023
[22] Agostinelli S et al. (Geant4 collaboration) 2003 Nucl. Instrum. Meth. A506 250
[23] Allison J, Amako K, Apostolakis J, Araujo H, Dubois P et al. (Geant4 collaboration) 2006 IEEE Trans. Nucl.Sci. 53 270
[24] Selvaggi M 2014 J. Phys. Conf. Ser. 523 012033
[25] Di Simone A 2016 J. Phys. Conf. Ser. These proceedings
[26] Apostolakis J et al. 2015 J. Phys. Conf. Ser. 608 012023
[27] Amadio G et al. 2015 J. Phys. Conf. Ser. 664 072006
[28] Zimmermann M (ALICE) 2015 J. Phys. Conf. Ser. 608 012019 (Preprint 1502.06381)
[29] Tsaregorodtsev A et al. 2010 J. Phys. Conf. Ser. 219 062029
[30] Chen X et al. 2016 CERN Analysis Preservation: A Novel Digital Library Service to Enable Reusable and Reproducible Research (Springer) pp 347–356 ISBN 978-3-319-43997-6
[31] Clemencic M et al. 2016 J. Phys. Conf. Ser. These proceedings
[32] Mazurov A and Couturier B 2016 J. Phys. Conf. Ser. These proceedings