The LHCb Software and Computing Upgrade towards LHC Run 3

S Roiser[1] and C Bozzi[1,2]
on behalf of the LHCb Computing Project

[1] European Organization for Nuclear Research (CERN), Geneva, Switzerland
[2] Sezione INFN di Ferrara, Ferrara, Italy

E-mail: stefan.roiser@cern.ch, concezio.bozzi@cern.ch

Abstract. LHCb is planning major changes for its data processing and analysis workflows for LHC Run 3. Removing the hardware trigger, a software only trigger at 30 MHz will reconstruct events using final alignment and calibration information provided during the triggering phase. These changes pose a major strain on the online software framework which needs to improve significantly. The foreseen changes in the area of the core framework include a re-design of the event scheduling, introduction of concurrent processing, optimizations in processor cache accesses and code vectorization. Furthermore changes in the areas of event model, conditions data and detector description are foreseen. The changes in the data processing workflow will allow an unprecedented amount of signal events to be selected and therefore increase the load on the experiments simulation needs. Several areas of improvement for fast simulation are currently being investigated together with improvements needed in the area of distributed computing. Finally the amount of data stored needs to be reflected in the analysis computing model where individual user analysis on distributed computing resources will become inefficient. This contribution will give an overview of the status of those activities and future plans in the different areas from the perspective of the LHCb computing project.

1. Introduction

The LHCb experiment will be upgraded for data taking in Run 3 and after [1]. Tu fully profit from the increase in instantaneous luminosity from 4×10^{32} to $2 \times 10^{33} cm^{-2} s^{-1}$ 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 [2]. In summary, output bandwidths between 2 and 5 GB/s are expected, to be compared with 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. It is therefore mandatory to study how the current model can be changed to sustain this scenario.

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 processing, and to implement data streaming as early as possible in the computing model.

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 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 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 parameterized simulations are employed where possible. Section 3 summarizes the work plan for these activities. A summary is given in Section 5.

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.

2.1. Framework

The data processing model of the `Gaudi` framework [5, 6] used in LHCb currently implies sequentially processing of events. 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, *i.e.* 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.

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.

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. and full photon reconstruction in the RICH detectors. Preliminary measurements of the performance show a good scalability in terms of multi-threading, a very low and scalable memory footprint. Fig 1 shows a comparison of memory usage where the multi-threaded jobs uses a factor 40 less memory than the multi-processing one.

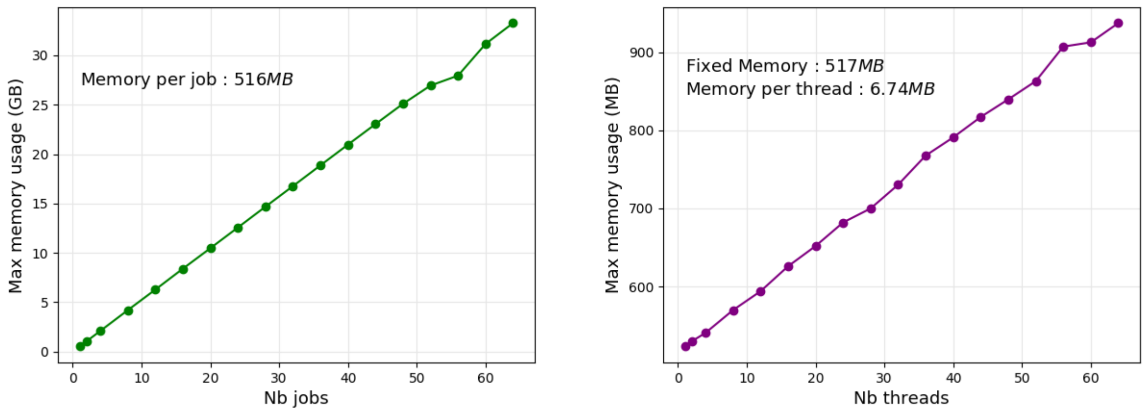

Figure 1. Memory consumption for multi-job and multi-thread approaches

2.2. Event Model

The LHCb event data model [9] 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.) and to use single precision where possible.

2.3. Detector Description and Conditions Database

The LHCb Detector Description (`LHCbDD` [10]) 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 [11] has been investigated as a replacement. It has been shown[12] that using `DD4Hep` from `Gaudi` is relatively easy. The full LHCb geometry has been automatically converted and tested. In addition, a geometry where the detailed structure of the LHCb detector is averaged out over a small number of elements, has been defined which e.g. improves the throughput for algorithms such as the Kalman filter.

The LHCb conditions data are currently stored in a database managed with the `Cool/Coral` libraries [13], developed by CERN/IT in collaboration with LHCb and ATLAS. The level of thread-safety of the `Cool/Coral` library is unclear. Moreover, the XML format used for

		SSE4		AVX2	
		time (s)	Speedup	time (s)	Speedup
double	scalar	233.462		228.752	
	vectorized	122.259	1.90	58.243	3.93
float	scalar	214.451		209.756	
	vectorized	55.707	3.85	26.539	7.90

Table 1. Performance of vectorized Rich photon reconstruction

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`. A prototype to manage the conditions backend with `git` is also available, that offers better performance and easier maintenance.

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 [14, 15]. An example for performance improvement in the RICH photon reconstruction via code vectorsation is shown in Table 1. However, a large investment in code rewrite should be taken into account and a cost-benefit analysis should be performed before taking decisions, as well as the reproducibility of results obtained on these alternative architectures.

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 [16, 17]. 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.

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 parallelized simulation frameworks (multi-threading, multi-processor) and of geometries of different complexity.

The flexibility of the LHCb Simulation framework, `Gauss` [18], allows to implement, in addition to the full `Geant4` simulation [19, 20], a variety of predefined safe and easy to use alternative configurations. Many options are possible including simulating only part of an event, replacing a full `Geant4` simulation with faster versions for a given detector or disabling specific processes (*e.g.* Cherenkov effect in RICH detectors), stopping particles at a specific stage, reducing the geometry, 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 [21].

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`.

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` [22]. 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`. A fully vectorized geometry package, `VecGeom`, is being developed in the context of `GeantV` [23], the vectorized version 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 with simpler settings but in an environment very similar to that of LHCb.

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. 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 optimization.

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. On average, the μDST occupies ten times less disk space than a DST. It is known that, although several analyses cannot be performed on μDST, full event information is still not needed in most cases. Therefore it is necessary to investigate new data formats in between μDST and DST where, for example, only a cone of tracks around the selected candidate is stored such that storage usage is optimized without compromising the physics analysis.

4.3. Centralised ntuple production and alternative approaches
It is foreseen that ROOT [24] ntuples will still be heavily used to perform data analysis which are produced by users in an unscheduled activity by running over `Turbo` or stripping output. The possibility to have a scheduled, centralized ntuple production is being investigated, based on the work done within ALICE [25]. These productions would be centrally managed in a timescale of weeks. It should also be noted that (μ)DSTs can also be directly used for physics analysis, but this requires at the moment the entire LHCb software stack.

4.4. Distributed computing
Infrastructures such as `Dirac` [26] will still be the main tool through which data is processed and distributed. The modular architecture of `Dirac` allows for adiabatic improvements of components, 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. 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 fast simulations 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, M. Cattaneo, R. Cenci, P. Charpentier, M. Clemencic, A. Contu, G. Corti, B. Couturier, C. Haen, 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] Michielin E (LHCb) 2016 *PoS* **ICHEP2016** 996. 4 p URL `https://cds.cern.ch/record/2287601`

[4] Martinelli M and Collaboration L 2017 *Journal of Physics: Conference Series* **898** 032039 URL `http://stacks.iop.org/1742-6596/898/i=3/a=032039`

[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] 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`

[10] Ponce S, Mato Vila P, Valassi A and Belyaev I 2003 *eConf* **C0303241** THJT007 (*Preprint* `physics/0306089`)

[11] `https://github.com/AIDASoft/DD4hep`

[12] Clemencic M and Karachaliou A 2015 *J. Phys. Conf. Ser.* **664** 072012

[13] Valassi A, Basset R, Clemencic M, Pucciani G, Schmidt S A and Wach M 2008 COOL, LCG conditions database for the LHC experiments: Development and deployment status *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

[14] Faerber C 2017 *Journal of Physics: Conference Series* **898** 032044 URL `http://stacks.iop.org/1742-6596/898/i=3/a=032044`

[15] Gallorini S, Lucchesi D, Gianelle A, Amerio S and Corvo M 2017 *Journal of Physics: Conference Series* **898** 032029 URL `http://stacks.iop.org/1742-6596/898/i=3/a=032029`

[16] Prez D H C 2017 *Journal of Physics: Conference Series* **898** 032052 URL `http://stacks.iop.org/1742-6596/898/i=3/a=032052`

[17] Stahl M 2017 *Journal of Physics: Conference Series* **898** 042042 URL `http://stacks.iop.org/1742-6596/898/i=4/a=042042`

[18] Clemencic M *et al.* 2011 *J. Phys. Conf. Ser.* **331** 032023

[19] Agostinelli S *et al.* (Geant4 collaboration) 2003 *Nucl. Instrum. Meth.* **A506** 250

[20] Allison J, Amako K, Apostolakis J, Araujo H, Dubois P *et al.* (Geant4 collaboration) 2006 *IEEE Trans.Nucl.Sci.* **53** 270

[21] Selvaggi M 2014 *J. Phys. Conf. Ser.* **523** 012033

[22] Apostolakis J *et al.* 2015 *J. Phys. Conf. Ser.* **608** 012023

[23] Amadio G *et al.* 2015 *J. Phys. Conf. Ser.* **664** 072006

[24] Brun R and Rademakers F 1997 *Nucl. Instrum. Meth.* **A389** 81–86

[25] Zimmermann M (ALICE) 2015 *J. Phys. Conf. Ser.* **608** 012019 (*Preprint* `1502.06381`)

[26] Tsaregorodtsev A *et al.* 2010 *J. Phys. Conf. Ser.* **219** 062029