EPJ Web of Conferences **214**, 05014 (2019) https://doi.org/10.1051/epjconf/201921405014
CHEP 2018

Improvements to the LHCb software performance testing infrastructure using message queues and big data technologies

Maciej Szymański[1,2,*] and *Ben* Couturier[2,**] on behalf of the LHCb collaboration

[1]University of Chinese Academy of Sciences, Beijing, China
[2]CERN, European Organization for Nuclear Research, Geneva, Switzerland

Abstract. Software is an essential component of High Energy Physics experiments. Due to the fact that it is upgraded on relatively short timescales, software provides flexibility, but at the same time is susceptible to issues introduced during development process, thus mandating systematic testing. We present recent improvements to LHCbPR, the framework implemented at LHCb to measure physics and computational performance of complete applications. This infrastructure is essential for keeping track of the optimisation activities related to the upgrade of computing systems which is crucial to meet the requirements of the LHCb detector upgrade for the next stage of data taking of the LHC. Latest developments in LHCbPR include application of messaging system to trigger the tests right after the corresponding software version is built within LHCb nightly builds infrastructure. We will also report on the investigation of using big data technologies in LHCbPR. We have found that using tools such as Apache Spark and Hadoop Distributed File System may significantly improve the functionality of the framework, providing an interactive exploration of the test results with efficient data filtering and flexible development of reports.

1 Introduction

The LHCb experiment will be upgraded, starting from 2019, for the next stage of data taking of the Large Hadron Collider (LHC) [1]. It is planned that the inelastic collision rate of 30 MHz will be processed by the full software trigger. One expects that an output bandwidth of up to 10 GB/s from the online trigger system to the offline computing system will be required in order to fully exploit the capabilities of the LHCb detector. Current data storage capacity and computing power are not sufficient to process data at this rate. This challenges several areas of the LHCb software and computing infrastructure [2]. In particular the infrastructure for the software performance measurements is of special importance to keep track of the optimisation activities related to the upgrade of computing systems. The goal of such a framework is to automate the process of the execution and scheduling of the tests as well as to collect the metrics of interest and store them for the future reference. The crucial feature of the system is to ensure controlled conditions to obtain a reliable performance baseline.

*e-mail: Maciej.Szymanski@cern.ch
**e-mail: Ben.Couturier@cern.ch

As a result, one can easily inspect any changes in the software behaviour introduced in subsequent software versions. Furthermore, it is convenient to be able to compare results of the tests across various compilers and architectures. It should be also emphasised that the project being the subject of this work aims at providing the tools to analyse not only resource consumption such as CPU time or memory usage, but also to study the performance of the physics quantities derived from running physics applications. LHCbPR solves also the common issue that tests of software performance are ran only by experts which requires specific knowledge to setup, run and understand the test. In addition, such workflow is resource consuming, inefficient due to manual comparison, and not useful for the whole collaboration if results are not available publicly. Another strength of the proposed solution is that since LHCbPR tests are running on dedicated machines, they can run much longer than unit tests (making it suitable for use cases which require processing a lot of events) and provide more information than a simple boolean value.

The LHCb Performance and Regression framework was introduced in [3] and [4]. The following paper elaborates on the recent developments of this tool. In section 2 the design of the infrastructure and current status of the system are described. Section 3 presents the prototype of the application of big data tools for the LHCbPR. The most common use cases of the LHCbPR are reported in section 4.

2 Infrastructure

The LHCbPR framework is organised following a microservice architecture. The main building components are: (i) back-end providing API service to retrieve the results of the tests from the database, (ii) web front-end enabling to browse, compare and plot test results, (iii) handlers responsible for parsing the output of the test results. Back-end and front-end modules are running as docker [5] containers and are combined using docker-compose [6].

The workflow in the LHCbPR system is presented in figure 1. The tests are triggered by user requests and messages coming from the LHCb continuous integration system [7, 8]. In the former case, the user specifies the application, its version, the option file stored in the dedicated repository and the handler module. The message with such information is sent to the queue of the tests, which are periodically consumed by one of the jobs (implemented in python [9] and bash [10]) in the LHCb instance of the Jenkins [11] infrastructure. In the latter case, once each application for a given platform (defined by the compiler, architecture and operating system) is compiled, the message is sent to the queue of builds which is checked against the lists of tests scheduled in XML file. The implementation of the message queue is done using the RabbitMQ message broker [12]. Such a solution allows to efficiently use CPU time of the build and test machines as the test jobs are scheduled to run on dedicated nodes as soon as the corresponding software is built. In particular benchmarks measuring resource consumption are run on machines where the unwanted load is minimised. As a next step of the LHCbPR workflow, output of the tests is parsed by the handlers. Those python modules produce zipped JSON [13] files with the metrics of interest and optionally ROOT [14] files. Furthermore, the zip files are uploaded to Dirac Storage Element [15] and then imported into MySQL [16] database using the back-end implemented using the Django framework [17]. The outcome is available in an AngularJS [18]-based dashboard. In the meantime, notifications about new results are sent to a Mattermost [19] channel for the interested users. The system allows to define for each monitored metric a corresponding threshold value which may trigger an alarm when exceeded. However due to high number of tests and rapidly evolving software, it appeared to be too difficult to maintain and is not commonly used.

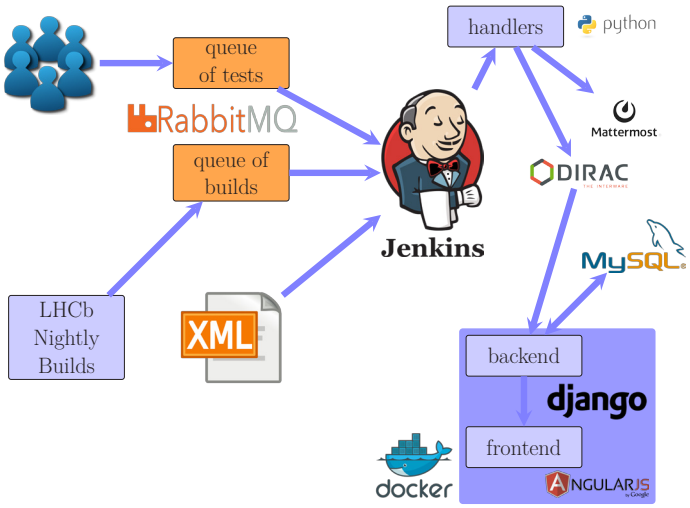

Figure 1. LHCbPR workflow

Currently, around 70 tests are defined for LHCb applications devoted for reconstruction, simulation and triggering. Single tests are running from several minutes up to 10 hours [1], depending on their goals. About 50 tests are running daily which produce tens of MB of data. After 2 years of operation, more than 10 GB were collected.

3 Enhancement with big data technologies

Numerous technologies emerged to support the analysis of the big data, understood in terms of its volume, variety and velocity. Such tools are especially useful for data exploration and creating the data warehouses. One of the distinctive examples is the Hadoop ecosystem [20]. The volume of the LHCbPR data cannot be considered as large-scale yet, however its variety makes Hadoop an interesting framework to be applied for the analysis of the results of the LHCb software tests. In particular, it enables to perform the interactive analysis of the data and therefore is more flexible comparing with the current approach based on the statically generated dashboard. Hence, the access to data is easier and faster. Indeed, short turn-around is desirable by data analysts. Moreover, combining those tools with shared notebooks makes such system perfect for collaboration and reproducibility.

The diagram presenting the prototype for the integration of the LHCb framework with the Hadoop ecosystem is shown in figure 2. It is based on Hadoop service provided by the CERN IT group [21]. In this setup the results from the tests are stored on the EOS storage [22] apart from the LHCbPR database. Dedicated scripts merge the JSON files daily and copy them into the Hadoop Distributed File System [23] (HDFS). Afterwards the data is converted into the Apache Parquet [24] format partitioned by application name and the option file which significantly reduces the time needed for data filtering. A compression rate with respect to the JSON files was found to be of the order of 16. Parquet format was chosen since it provides fast data ingestion and random data access, and scalability [25]. For data analysis, Apache Spark [26] processing engine is used through the integration with

[1]This limit comes from the settings of our Jenkins job. It was chosen to prevent the excessive usage of the machines on which the tests are running. The limit is based on the experience gained on the use cases in LHCb.

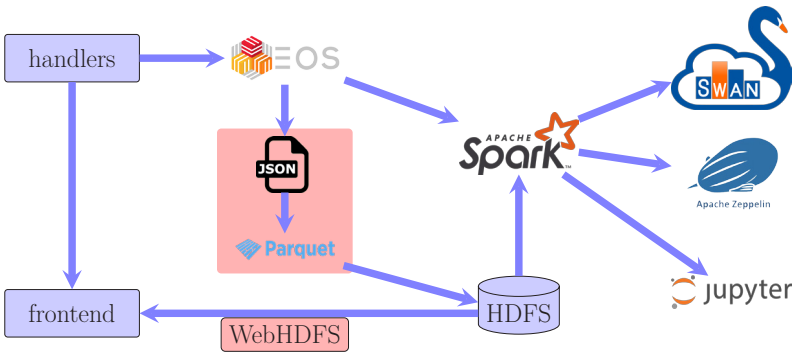

Figure 2. Integration of LHCbPR with Hadoop ecosystem

notebooks, in particular Apache Zeppelin [27] and SWAN [28]. Some of the LHCbPR tests produce ROOT files as well as JSON files. To read them, the spark-root [29] package was investigated and found to be useful for that case. It is planned also to investigate WebHDFS protocol [30] to read LHCbPR data in web dashboard in parallel to approach based on HDFS and notebooks. It should be pointed out that it may not be possible to transform analysis modules in the existing web front-end into SWAN notebooks preserving the same usability and functionality. Therefore, our goal is to support both methods of data analysis in LHCbPR. Typical analysis in SWAN is based on the PySpark [31] module making also use of matplotlib library [32]. There is also a way to introduce some interactivity by applying widgets provided by Jupyter notebooks [33].

4 Use cases

The flagship analysis performed in LHCbPR framework is to plot given measurements as a function of the software version. Memory usage of the application, CPU time spent by a given algorithm, reconstruction efficiency, or throughput (number of processed events per second by reconstruction code) are most common examples of the monitored metrics. Figure 3 presents CPU time spent in the event loop by a benchmark job which is a simulation of inclusive b-events in proton-proton collisions at 13 TeV performed using the Gauss simulation application [34]. Labels on the x-axis correspond to different versions of Gauss. Decrease of the time for versions 4 and 10 is due to changes in the Ring Imaging Cherenkov (RICH) detector [35] code, specifically to speed it up which were introduced in corresponding software versions. This indicates the power of the LHCbPR framework as in an automated way one can observe any variations in the code output introduced by e.g. Gitlab's merge request [36], new external libraries or Monte Carlo generators.

Other use cases of the LHCbPR framework include rate and throughput tests for the High Level Trigger [37], and simulation validation [38]. Furthermore, there are also automated tests employing code profiling tools such as perf [39] and IgProf [40], which make it possible to analyse where the program spends most time, and to investigate memory leaks.

5 Summary

Monitoring of the software is an essential tool in large scientific projects such as LHCb. The LHCbPR system has already shown to be a versatile framework useful for the whole

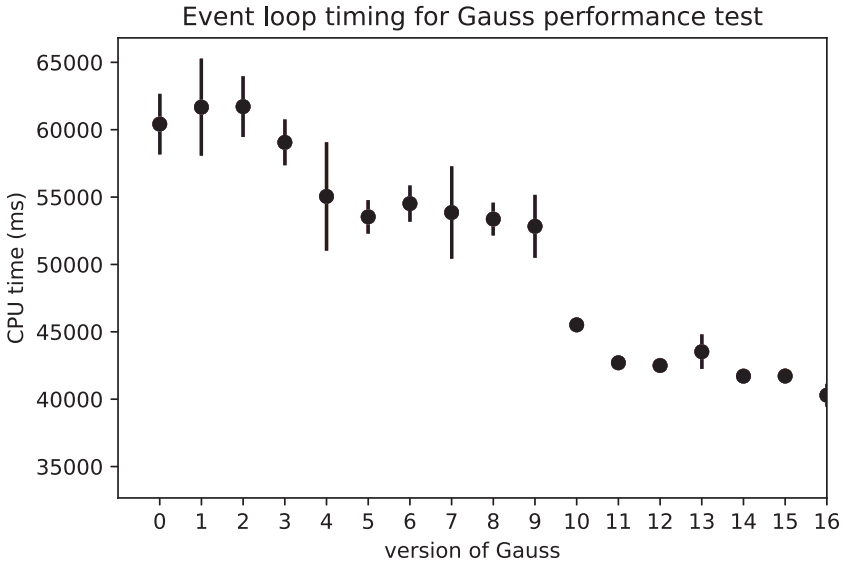

Figure 3. CPU time spent by the simulation application as function of the software version.

collaboration, improving the monitoring and control of the software developed for the LHC upgrade era. The LHCbPR project is not coupled to the LHCb software stack making it suitable to be used by other projects dealing with large and rapidly evolving code base. In addition, this paper indicates that big data tools appear to be promising for the integration with LHCbPR since they enable to efficiently create customisable reports on the results of the tests developed in shared notebooks making them reproducible and convenient to collaborate.

References

[1] I. Bediaga *et al.* [LHCb Collaboration], "Framework TDR for the LHCb Upgrade : Technical Design Report," CERN-LHCC-2012-007, LHCb-TDR-12.

[2] The LHCb Collaboration, CERN, "Upgrade Software and Computing," CERN-LHCC-2018-007. LHCB-TDR-017, https://cds.cern.ch/record/2310827,

[3] B. Couturier, E. Kiagias and S. B. Lohn, "Systematic profiling to monitor and specify the software refactoring process of the LHCb experiment," J. Phys. Conf. Ser. **513** (2014) 052020. doi:10.1088/1742-6596/513/5/052020

[4] A. Mazurov, B. Couturier, D. Popov and N. Farley, "Microservices for systematic profiling and monitoring of the refactoring process at the LHCb experiment," J. Phys. Conf. Ser. **898** (2017) no.7, 072037. doi:10.1088/1742-6596/898/7/072037

[5] Docker [software], https://www.docker.com/ [accessed 2018-10-19]

[6] Docker Compose [software] https://docs.docker.com/compose/ [accessed 2018-10-19]

[7] M. Clemencic and B. Couturier, "LHCb Build and Deployment Infrastructure for run 2," J. Phys. Conf. Ser. **664** (2015) no.6, 062008. doi:10.1088/1742-6596/664/6/062008

[8] S.-G. Chitic, "LHCb continuous integration and deployment system: a message based approach", CHEP 2018

[9] Python [software], https://www.python.org/ [accessed 2018-10-19]

[10] Bash [software], https://www.gnu.org/software/bash/ [accessed 2018-10-19]
[11] Jenkins [software], https://jenkins.io/ [accessed 2018-10-19]
[12] RabbitMQ [software], https://www.rabbitmq.com/ [accessed 2018-10-19]
[13] JSON [software], https://www.json.org/ [accessed 2018-10-19]
[14] I. Antcheva *et al.*, "ROOT: A C++ framework for petabyte data storage, statistical analysis and visualization," Comput. Phys. Commun. **180** (2009) 2499 doi:10.1016/j.cpc.2009.08.005 [arXiv:1508.07749 [physics.data-an]].
[15] A. Tsaregorodtsev *et al.*, J. Phys. Conf. Ser. **119** (2008) 062048. doi:10.1088/1742-6596/119/6/062048
[16] MySQL [software], https://www.mysql.com/ [accessed 2018-10-19]
[17] Django [software], https://www.djangoproject.com/ [accessed 2018-10-19]
[18] AngularJS [software], https://angularjs.org/ [accessed 2018-10-19]
[19] Mattermost [software], https://mattermost.com/ [accessed 2018-10-19]
[20] Hadoop [software], https://hadoop.apache.org/ [accessed 2018-10-19]
[21] https://information-technology.web.cern.ch/services/Hadoop-Service/ [accessed 2018-10-19]
[22] http://information-technology.web.cern.ch/services/eos-service/ [accessed 2018-10-19]
[23] HDFS [software], https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html/ [accessed 2018-10-19]
[24] Apache Parquet [software], https://parquet.apache.org/ [accessed 2018-10-19]
[25] Z. Baranowski, L. Canali, R. Toebbicke, J. Hrivnac and D. Barberis, "A study of data representation in Hadoop to optimize data storage and search performance for the ATLAS EventIndex," J. Phys. Conf. Ser. **898** (2017) no.6, 062020. doi:10.1088/1742-6596/898/6/062020
[26] Apache Spark [software], https://spark.apache.org/ [accessed 2018-10-19]
[27] Apache Zeppelin [software], https://zeppelin.apache.org/ [accessed 2018-10-19]
[28] Danilo Piparo and Enric Tejedor and Pere Mato and Luca Mascetti and Jakub Moscicki and Massimo Lamanna "SWAN: A service for interactive analysis in the cloud", Future Generation Computer Systems, volume 78, 1071 - 1078, 2018, issn: 0167-739X, doi: https://doi.org/10.1016/j.future.2016.11.035, http://www.sciencedirect.com/science/article/pii/S0167739X16307105
[29] Viktor Khristenko and Jim Pivarski, "diana-hep/spark-root: Release 0.1.14", doi:10.5281/zenodo.1034230, https://doi.org/10.5281/zenodo.1034230
[30] WebHDFS [software], https://hadoop.apache.org/docs/r1.0.4/webhdfs.html/ [accessed 2018-10-19]
[31] PySpark [software], http://spark.apache.org/docs/2.2.0/api/python/pyspark.html/ [accessed 2018-10-19]
[32] Matplotlib [software], https://matplotlib.org/ [accessed 2018-10-19]
[33] Jupyer [software], http://jupyter.org/ [accessed 2018-10-19]
[34] Gauss [software], http://lhcbdoc.web.cern.ch/lhcbdoc/gauss/ [accessed 2018-10-19]
[35] M. Adinolfi *et al.* [LHCb RICH Group], Eur. Phys. J. C **73** (2013) 2431 doi:10.1140/epjc/s10052-013-2431-9 [arXiv:1211.6759 [physics.ins-det]].
[36] Gitlab [software], https://about.gitlab.com/ [accessed 2018-10-19]
[37] R. Currie, "Monitoring LHCb Trigger developments using nightly integration tests and a new interactive web UI", CHEP 2018
[38] D. Popov, "Testing and verification of the LHCb Simulation", CHEP 2018

[39] Perf [software], https://perf.wiki.kernel.org/index.php/Main_Page/ [accessed 2018-10-19]

[40] Eulisse and Tuura, "IgProf profiling tool", Proceedings CHEP04, Computing in High Energy Physics,