EPJ Web of Conferences **214**, 05042 (2019) https://doi.org/10.1051/epjconf/201921405042
CHEP 2018

Monitoring LHCb Trigger developments using nightly integration tests and a new interactive web UI

Robert Currie[1,*] and *Conor* Fitzpatrick[2,**] on behalf of the LHCb Collaboration

[1]Edinburgh University, James Clerk Maxwell Building, Peter Guthrie Tait Road, Edinburgh, EH9 3FD
[2]Laboratory for High Energy Physics (LPHE), EPFL-SB-IPHYS-LPHE BSP - Cubotron CH-1015 Lausanne

Abstract.
The LHCb Performance Regression (LHCbPR) framework allows for periodic software testing to be performed in a reproducible manner. LHCbPR provides a JavaScript based web front-end service, built atop industry standard tools such as AngularJS, Bootstrap and Django. This framework records the evolution of tests over time allowing for this data to be extracted for end-user analysis.
The LHCbPR framework has been expanded to integrate the nightly testing and profiling. These developments allow for key performance metrics within the Trigger software to be monitored over time. Additionally, tests of the full physics reconstruction have been integrated into LHCbPR. These allow for tracking the effect that optimization work has on physics reconstruction performance.
This presentation will describe the integration of these tests into LHCbPR as well as describing the structure and new features developed for the frontend web service.

1 Introduction

The LHCb experiment in LHC Run3 will be migrating toward a software based Higher Level Trigger from a Hardware based solution. In doing this the experiment will be increasing the rate that data is written out to disk from 700Mb/s to 2-5Gb/s. One of the advantages that this offers is the ability to rapidly deploy new versions of the HLT software whilst collecting data. This allows the trigger to adaptively change physics selection criteria as the experimental program evolves. To ensure new versions of the software are behaving correctly it is important to monitor the development of the software. In order to do this nightly tests are run and results are stored within the LHCb Performance Regression framework for comparison and monitoring. These nightly tests are designed to track and record the computing and physics performance.

Recent work on LHCbPR has been focussed on enhancements to allow the tracking of results from the LHCb HLT software. This work has resulted in the development of a new web interface focussing on user interaction with data that has previously been collected. Additionally work has been done to expanding different components of the LHCbPR framework to efficiently handle large amounts of numerical data.

*e-mail: rcurrie@cern.ch
**e-mail: conor.fitzpatrick@cern.ch

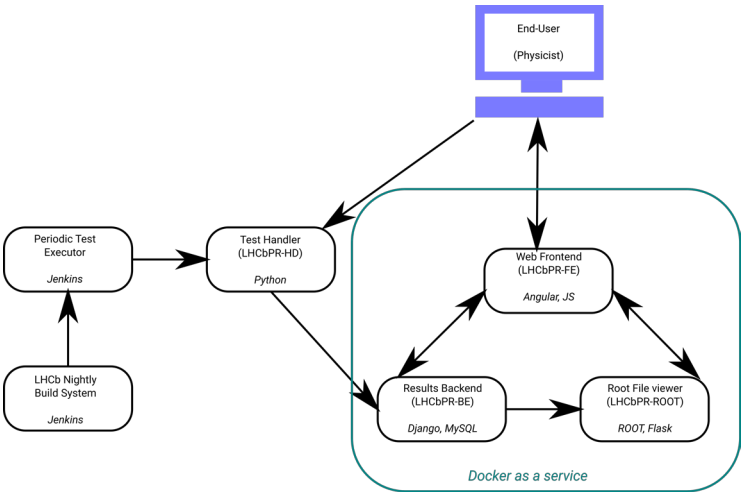

Figure 1. Components of the LHCbPR framework. The main components of this framework are:
LHCbPR-BE, this component manages the test configurations and data collected.
LHCbPR-FE, this component provides an interactive web-UI for end-users.
LHCbPR-HD, this component contains users scripts for parsing test results.
LHCbPR-ROOT, this component provides access to a ROOTJS instance for parsing ROOT files.

2 An overview of LHCbPR

The LHCbPR is a web framework built atop the LHCb nightly build system[1]. This framework is composed of multiple distinct components as shown in Figure 1.

The LHCb nightly build system makes extensive use of a central software build service which is responsible for managing the nightly builds of the LHCb software architecture for multiple platforms and code branches. Templated descriptions of the nightly tests and their requirements are stored in the nightly build system. The actual configuration of the tests which are run each night are constructed through populating templates using information in the LHCbPR-BackEnd (LHCbPR-BE) as in Figure 1.

Once the tests have completed the results are parsed using scripts from LHCbPR-HD (Figure 1). From this the output is uploaded to LHCbPR-BE and LHCbDirac[2]. The results of these tests are then available to the end user through either the LHCbPR-FrontEnd (LHCbPR-FE in Figure 1) or can be directly accessed through the LHCbPR-BE.

3 HLT nightly test Integration

The HLT nightly tests are designed to monitor the performance of the HLT software from both a computing and physics perspective. These tests operate by running the HLT selection over a given sample of MC data which has been selected to test the performance of the HLT from multiple perspectives. Information collected from these tests comes in multiple formats which often have a unique set of requirements.

Parameters that have been chosen to be monitored using the LHCbPR system are the rate and time taken of various physics selections in software. The computing parameters of interest are the relative CPU and memory consumption of various components within the executable.

In order to be able to use the LHCbPR system to monitor this data it had to be enhanced to interactively display large amounts of numerical data. This interface would allow a user to explore the history of a parameter over a period of time as well as to compare values between selected different builds.

4 Enhancements made to the LHCbPR-BE framework

The LHCbPR framework provides access to the saved results of nightly tests through an API built on the Django[3] rest framework[4]. Initially this API was user orientated and returned data in large JSON objects. These objects typically contained both the information of interest as well as metadata including object relations.

This was designed to allowing people to easily navigate through the data stored in the LHCbPR-BE. Unfortunately this approach added a significant overhead when used in designing an interactive web framework.

To allow for more user interaction with the data through the LHCbPR-FE there were an additional requirements placed on being able to filter and sort the data from LHCbPR-BE. Given recent performance improvements in web browsers it was chosen to provide this functionality through the LHCbPR-FE component. Redesigning the LHCbPR-FE this way added the requirement that LHCbPR-BE be able to efficiently serve the results from a user query on demand.

Adding new API calls to the LHCbPR-BE allowed for the data to be parsed in a more efficient way and also simplified the flow of data through LHCbPR-BE system. These new API calls were written with performance in mind. Higher performance was achieved by assembling the full database queries ahead of execution minimizing database interaction, and changing the format of LHCbPR-BE responses to reduce the amount of traffic between LHCbPR-BE and LHCbPR-FE.

5 Interactive LHCbPR-FE development

The key requirements placed on the LHCbPR system are that the results be reproducible and the user interface allow for user interaction. With this in mind a new front-end for the LHCbPR system has been developed using Clarity [5] and Typescript [6]. There have been strong advantages to developing a web-UI atop industry standard tools such as these and that has been to improve the speed of deployment and stability of the releases.

An example of the web interface which updates on demand from the LHCbPR-FE component is shown in Figure 2. This interface allows users to interact with data in a graphical and tabular manner to explore many different test results. The results from a nightly test are presented in this way to the browser to allow for the user to manipulate the data on demand to meet their requirements.

LHCbPR-FE makes use of tracked internal states to allow for a user to save, share and compare plots they've generated using the web framework. Tracking the users interaction this way made it possible to provide reliable and reproducible behaviour important for collaborative working. As well as providing a more user-friendly web-UI the new LHCbPR-FE interface makes more extensive use of the query based enhancements made to the LHCbPR-BE.

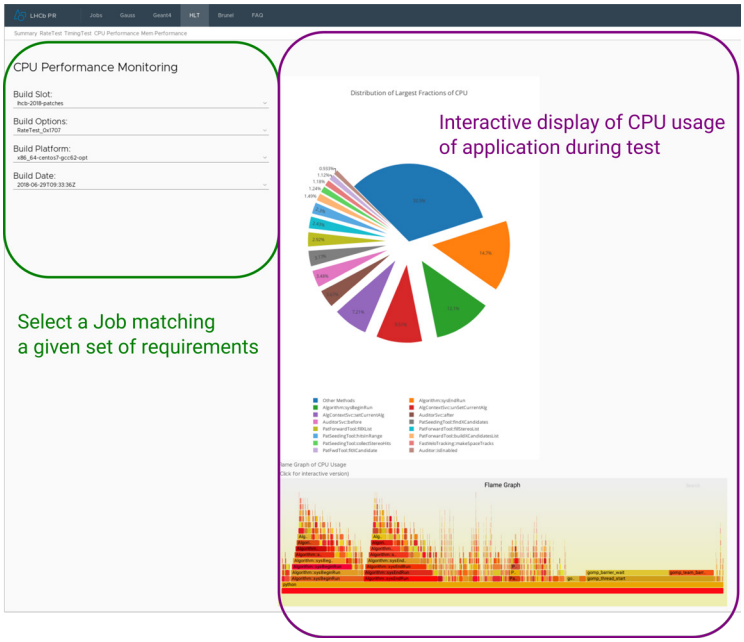

Figure 2. An Example of the interactive web user interface for the LHCbPR system. This is an example of the breakdown of the CPU usage of a nightly test.

6 Conclusion

The LHCbPR framework has proven itself to be highly flexible and expandable, allowing for the API to be expanded to meet additional demands. Recent enhancements to the LHCbPR framework have resulted in a system which can interactively display large amounts of numerical data in an intuitive way. The integration of the HLT nightly tests into this system allows for the performance of HLT software from both a physics and computing perspective to be tracked and monitored.

References

[1] M. Clemencic and B. Couturier, *A New Nightly Build System for LHCb*, J.Phys.Conf.Ser. **513**, 052007, (2014)

[2] F. Stagni and P. Charpentier and .R Graciani and A. Tsaregorodtsev and J. Closier and Z. Mathe and M. Ubeda and A. Zhelezov and E. Lanciotti and V. Romanovskiy and K. D. Ciba and A. Casajus and S. Roiser and M. Sapunov and D. Remenska and V. Bernardoff and R. Santana and R. Nandakumar, *LHCbDirac: distributed computing in LHCb*, J.Phys.Conf.Ser. **396**, 032104, (2012)

[3] Django, *A high-level Python Web framework*, URL https://www.djangoproject.com

[4] Django Rest Framework, *A powerful and flexible toolkit for building Web APIs*, URL https://www.django-rest-framework.org

[5] Clarity Design System, *An open source design system*, URL https://vmware.github.io/clarity

[6] Typescript, *Javascript that scales*, URL https://www.typescriptlang.org/