Data preservation and reproducibility at the LHCb experiment at CERN

Ana Trišović

of

Newnham College

University of Cambridge

This dissertation is submitted for the degree of

Doctor of Philosophy

July 2018

Abstract

This dissertation presents the first study of data preservation and research reproducibility in data science at the Large Hadron Collider at CERN. In particular, provenance capture of the experimental data and the reproducibility of physics analyses at the LHCb experiment were studied.

First, the preservation of the software and hardware dependencies of the LHCb experimental data and simulations was investigated. It was found that the links between the data processing information and the datasets themselves were obscure. In order to document these dependencies, a graph database was designed and implemented. The nodes in the graph represent the data with their processing information, software and computational environment, whilst the edges represent their dependence on the other nodes. The database provides a central place to preserve information that was previously scattered across the LHCb computing infrastructure. Using the developed database, a methodology to recreate the LHCb computational environment and to execute the data processing on the cloud was implemented with the use of virtual containers. It was found that the produced physics events were identical to the official LHCb data, meaning that the system can aid in data preservation. Furthermore, the developed method can be used for outreach purposes, providing a streamlined way for a person external to CERN to process and analyse the LHCb data.

Following this, the reproducibility of data analyses was studied. A data provenance tracking service was implemented within the LHCb software framework GAUDI. The service allows analysts to capture their data processing configurations that can be used to reproduce a dataset within the dataset itself. Furthermore, to assess the current status of the reproducibility of LHCb physics analyses, the major parts of an analysis were reproduced by following methods described in publicly and internally available documentation. This study allowed the identification of barriers to reproducibility and specific points where documentation is lacking.

With this knowledge, one can specifically target areas that need improvement and encourage practices that would improve reproducibility in the future.

Finally, contributions were made to the CERN Analysis Preservation portal, which is a general knowledge preservation framework developed at CERN to be used across all the LHC experiments. In particular, the functionality to preserve source code from GIT repositories and DOCKER images in one central location was implemented.

Declaration

This dissertation is the result of my own work, except where explicit reference is made to the work of others. It has not been submitted for another qualification to this or any other university. This dissertation does not exceed the word limit for the respective Degree Committee.

Ana Trišović

Acknowledgements

Data preservation and reproducibility have not been the primary focus of high-energy physics in the previous years, and therefore, this project, which is the first such study at the Large Hadron Collider at CERN, would not be possible without numerous collaborators coming from physics, computer and information sciences. I have benefited from discussions with a number of people, and here I would like to acknowledge their help and ideas.

Thank you to my supervisors for overseeing this project. I would like to thank Ben Couturier for his ideas and his help, and I would like to thank Chris Jones for reading this thesis multiple times, teaching me about LHCb and for being patient with my odd English constructions.

I am indebted to Professor Val Gibson and Vladimir Vava Gligorov for giving me the opportunity to undertake this research at the University of Cambridge and CERN. I am grateful for their helpful suggestions about this thesis. Also, I have benefited from continuous encouragement and advice from Vava.

I would like to thank Agnieszka Dziurda for teaching me physics, answering my endless stream of questions about LHCb, suggesting ideas for this project and reading parts of this thesis. Thank you to Silvia Amario and Sebastian Neubert for their helpful suggestions and fruitful discussions.

I would like to thank Alison Tully, with whom I tackled the challenges of reproducible research by attempting to reproduce an LHCb analysis. Thank you to Jordi Garra Tico for his ideas, contributions to the analysis, and for reading this thesis. Thank you to Harry Cliff who has advised me in the initial stages of the analysis. Also, I would like to thank Tina Potter, my College mentor, for her encouragement, guidance and advice.

I would like to offer my gratitude to my team, LHCb Computing (LBC). Especially to Marco Cattaneo, Stefan Roiser, Gloria Corti, Zoltán Máthé, Philippe Charpentier, Christophe

Haen, Christoph Hasse and Federico Stagni for answering many of my questions about LHCb data. A special thank you to Marco Clemencic for helping me with my modest contribution to the Gaudi framework. And thank you to Sebastien Ponce for helping me whenever I needed help with my operating systems in our office. I would like to offer my special thanks to Rainer Schwemmer, Kazu Akiba and Christian Faerber, who have taken me out for drinks, but who also talked to me about my project. I would like to thank Vicor Coco for his ideas in the early (and final) stages of this project.

I would like to offer my very special thanks to Thomas F. J.-M. Pasquier and Matthew K. Lau. I very much enjoyed our (for me) afternoon (for you morning) meetings and fruitful discussions.

I would like to offer my deepest gratitude to Sünje Dallmeier-Tiessen who has been my supervisor, mentor and a role model. Thank you for your patience, guidance and support, and for allowing me to join the CERN Open Data and CERN Analysis Preservation (CAP) team. I would like to thank the CAP team with a special mention to Tibor Simko. Thank you to Xiaoli Chen for being my tester in the quest for reproducible research. I would like to thank Claudia Müller-Birn for advising me about my thesis plots. Also, thank you to Lukas Heinrich for his help.

I would like to thank my physics friends: Hannah Mary Evans, Thibaud Humair, Slavomira Stefkova, Nabarun Dev, Marcela Vitti, who I occasionally surveyed to learn more about common practices in physics analyses. I would particularly like to thank Åsmund Schiager Folkestad, my partner, for inspiring me, teaching me physics and reading this thesis. I would like to thank my friends and family who read this thesis or parts of it: Gerhard F. Rzehorz, Nefeli Kousi, Milena Bajić, Stephanie Van De Sandt and Zaga Trišović, with a special mention to William K. Balunas.

I would like to acknowledge financial support from the CERN Doctoral Student program, the Muir Wood studentship awarded by the Newnham College, Google Anita Borg Memorial scholarship, an award by the Cambridge fieldwork fund, and the Dositeja award by the Fund for Young Talents in Serbia.

I would like to thank my girlfriends: Marija Janković, Jelena Stojković, Ljiljana Rajačić and Nataša Dragović, and my other numerous friends for their continuous support and advice.

Finally, I would like to thank my family for their love, support and encouragement, without whom none of this would be possible. I dedicate this thesis to my parents.

Contents

Outline

Throughout the history of science, experiments have been traditionally done by small groups of people. The group would often perform the entire research process from the experiment design and data collection, to analysis and publications. In fields such as high-energy physics and astrophysics, this is not always the case anymore. Large scientific collaborations have emerged in these fields to advance their experiments more than ever before. The labour of running the experiments is divided into several groups of physicists, engineers and computer scientists, where each group has its own field of specialisation. Data collection is normally done on the experimental site as a joint effort of the collaboration, while data analysis can be done by physicists who, after the data is released, can be based anywhere around the globe.

In this thesis, I describe my study of data preservation and reproducibility starting from data collection to physics publications at the particle physics experiment LHCb at CERN. As data analyses have become convoluted and computational-resource intensive, many challenges in research reproducibility and scientific preservation have arisen. My goal is to study this data flow, identify obstacles and implement or suggest improvements in performing data analyses.

The data flow starts at the LHCb detector where the data is collected. To help understand the data, the high-energy physics (HEP) collaborations create a large amount of Monte Carlo simulations that mimic the data and the detector response. After data processing, the derived data and simulation are released to the physicists of the collaboration who conduct various data analyses to study and test the Standard Model of particle physics, thus creating a large number of physics publications. This data flow is shown in Figure 1 together with the outline of this thesis. My original work is described in detail in chapters 4, 5, 6, 7, 8 and 9. The contributions are complemented with GitHub repositories.

In chapter 1 I introduce the terminology related to the data preservation initiative. I explain the motivation for data preservation and challenges in research reproducibility in HEP

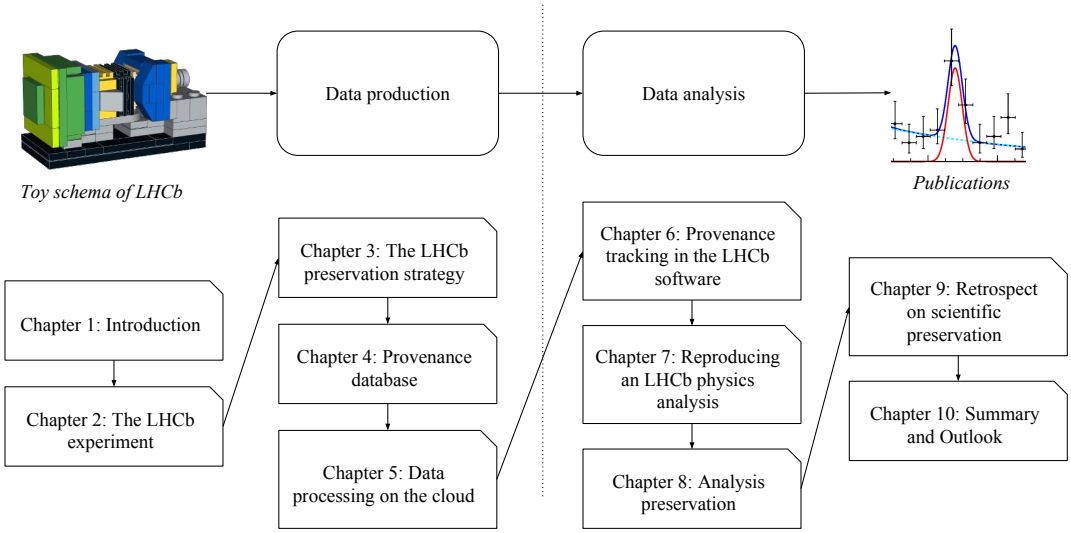


Figure 1: The thesis outline. The vertical line in the middle of the figure represents the division between what was done before the data reaches the analysts and after. The first part of the thesis describes *Data production*, which is a set of processes to prepare data for physics analyses. The second part addresses the challenges of preservation and reproducibility of *data analyses*.

and in science in general.

In chapter 2 I describe the components of the LHCb detector and talk about the changes that are going to be implemented in the near future. These upgrades of the experiment will result in changes in the data, which need to be considered for the preservation efforts.

In chapter 3 I introduce the long-term preservation strategy of LHCb. I classify the experiment resources into several components and discuss the challenges in their preservation.

In chapter 4 I explain how the experimental data is partially processed centrally at CERN and then distributed on the computing Grid for further processing. I design and implement a graph database to gather the provenance information about these processes [1, 2].

In chapter 5 I introduce a methodology to process physics data or recreate Monte Carlo simulations on the cloud using CERN's free and open source software [3]. To evaluate the outcome, I compare the events created on the cloud to the official events created on the Grid [4].

In chapter 6 I introduce a provenance tracking service in the LHCb software that collects and captures analysis job configurations within the output dataset [5]. I outline its implementation and functionality.

In chapter 7 I attempt to reproduce a physics analysis using only public and internally available documentation [6]. My goal is to identify reproducibility barriers and suggest improvements.

In chapter 8 I introduce the pillars of analysis preservation and the CERN Analysis Preservation framework. To demonstrate the preservation process, I use the analysis discussed in the previous chapter as an example. The aim of the chapter is to ensure reproducibility and reusability of the analysis.

In chapter 9 I present a survey on common practices in physics analyses and outline its implications. I discuss sociological aspects in adopting preservation and collaborative tools.

In chapter 10 I summarise my work and give an outlook for the future developments in data preservation and reproducibility in HEP.

Chapter 1

Introduction

The European Organisation for Nuclear Research (CERN) is an international organisation that operates what is currently the largest particle physics laboratory in the world. The laboratory is located at the French-Swiss border, near the city of Geneva in Switzerland. CERN was established in 1954, and since then it has been conducting physics programme ranging from studies on nuclear interactions to high-energy physics, and from antimatter to cosmic rays. As a result, the laboratory has been leading the advancement of fundamental physics research and detector technology.

Scientific preservation and reproducibility are one of the significant challenges in high-energy physics (HEP) and in science in general. CERN experiments are created to collect physics data over a lifetime of several decades. As data formats, detector hardware and software change over time, work must be done to access the old data effectively. The fact that it is a particular challenge to evaluate or reproduce previously published results creates a worrisome issue for research reproducibility at CERN. In this chapter, I introduce the details of this challenge and our motivation to overcome it.

1.1 Motivation

Data used in research testifies to the authenticity of the results and provides the ability for the research to be confirmed or improved. There is a high incentive to ensure the preservation of the data and research methods, and this can be demonstrated through several possible scenarios.

Particle physics experiments have become increasingly sophisticated, and they cannot be realistically repeated in the near future. This makes CERN data unique. Therefore, if

there is a need to evaluate a new physics theory, improve a current measurement or look for a new signal, it should be possible to reuse the data. If future data contradicts an important result, it is necessary to backtrack the methods and understand why. Furthermore, after an experiment has been completed or replaced, its data and software may no longer be maintained. Scientific preservation aids in extending the period for performing physics data analysis after an experiment has been completed or replaced.

The preservation of the data and research methods facilitates outreach, educational and scientific contributions to society. All major HEP experiments are publicly funded, and they follow open access policies for publishing their results. They are also obliged to provide outreach and educational material for the general public. The same policies apply to the research data, which typically needs to be published after an embargo period. However, publishing the data is not enough, and it should be followed by documentation to understand, and the software to utilise the data. Therefore outreach and open data efforts are strongly tied to data and software preservation. Ultimately, the data is genuinely preserved if it is publicly available, accessible and usable as open data.

Finally, preservation practices assist in research reuse by capturing the work and results of previous physics analyses. It is essential to develop a well-structured system, where scientists can find and reuse preserved research artefacts. The easier this process is, the faster the collaboration achieves its scientific goals [7].

1.2 Experiment and data lifecycle

To better understand the lifecycle of high-energy physics experiments and data, let us consider the study of the Higgs boson that was discovered in 2012 [8]. The particle was theoretically predicted back in the 1960s. The search for the experimental confirmation started by analysing the data collected at the experiments of the Large Electron Positron (LEP) collider at CERN. Novel results were obtained, such as the limits on the mass of the Higgs boson, but due to constraints of the collider design, results of high significance could not have been acquired [9]. Two experiments, ATLAS [10] and CMS [11], were designed and built to study a wide range of particle physics phenomena, one of the most important being the study of the Higgs boson. The design, construction and testing of the detectors took several decades before the experiments started collecting data in November 2009.

When two beams of particles collide, they can potentially produce new particles, such as

a Higgs boson. These collisions in particle physics are called *events*. During data collection, the data describing these events and newly created particles is recorded from the detectors and sent to storage.

There are commonly two stages of data processing that take place after data collection, as shown in Figure 1.1. The first stage is commonly referred to as *Data production*. This stage is managed centrally at CERN and is done by the on-site computing experts. In this stage, the raw events are reconstructed into physical quantities, such as particle mass, energy etc. After that, the data is made available to the physicists of the collaboration, who can be located at any affiliated institution around the globe, to perform analyses on the data. *Data analysis* represents a process of inspecting, cleaning and transforming the data. For this, physicists use highly specialised tools and algorithms. They compare the experimental data to the theoretical predictions with the goal of discovering useful physics quantities. The final result of these analyses is published in physics journals.

The transformation of data volume and complexity can be seen in Figure 1.1. After the data-taking, the size of the raw files is measured in petabytes (PB). During the data production and analysis, the data complexity and size are reduced, and the final output (plots, datasets etc.) are often on the order of magnitude of several megabytes (MB).

A number of different properties of a particle can be measured in a physics analysis. For instance, one can measure its mass, its average lifetime before it decays into other particles or the relative frequency in which the particle decays to some other specified combination of particles, also known as a branching fraction. In the context of searches for new physics or observing a new particle, physics results are often evaluated according to their statistical significance. This significance is typically measured in standard deviations of the normal distribution. Such evaluation introduces additional terminology in physics analyses. For example, analysis called *evidence* (for a particular signal) is at least 3σ significant. A confident result is the one called *observation* (*discovery*), as it is at least 5σ statistically significant.

In the context of the discovery of the Higgs boson, in July 2012, both ATLAS and CMS announced that they had observed *"a particle consistent with the Higgs boson"* [12]. Only a few months later, in August 2012, they reported an observation or "strong indications for the presence of a new particle that could be the Higgs boson" [8, 13], with the level of significance of 5σ.

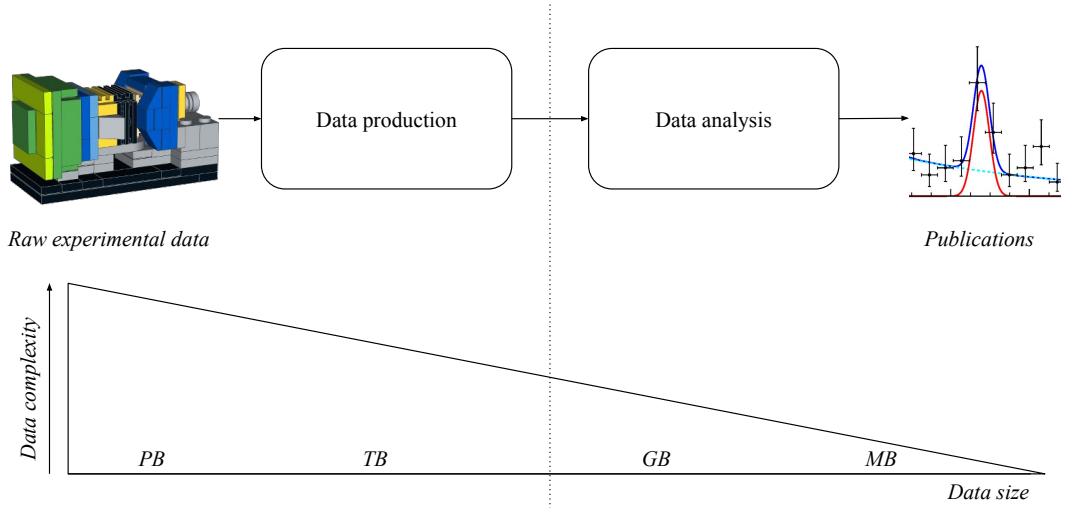


Figure 1.1: HEP data lifecycle and transformation of data volume and complexity in this process.

1.3 Reproducibility and repeatability

Reproducibility is one of the cornerstones of science, as it allows third-party scientists to verify and expand on research. I use the terminology from Vitek and Kalibera [14, 15] to define reproducibility and repeatability of a piece of work:

- **Reproducibility:** *is independent confirmation of a scientific hypothesis through reproduction by an independent researcher. The reproductions are carried out after a publication, based on the information in the paper and possibly some other information, such as datasets, published via scientific data repositories or provided by the authors on inquiry;*
- **Repeatability:** *the ability to re-run the exact same experiment with the same method on the same or similar system and obtain the same or very similar result.*

Scientific methods can include observations and measurements that are not repeatable by their nature. An example of this is a neutrino burst from the supernova SN1987A recorded at the Kamiokande II detector that lasted for 13s in 1987 [17]. Another example is an observation of the gravitational wave generated by the collision of two black holes that was recorded in 2015 at the Advanced LIGO detectors by the LIGO and Virgo Scientific Collaboration [18]. These observations are not repeatable because if they were missed or poorly measured, they could not have been measured again. Furthermore, any HEP measurement is not repeatable in the sense that it is impossible to build the "exact same experiment" and record the "exact same events". This is because the experimental setup has unique capabilities and its components are often custom built for specialised purposes. Also, this is because the physics events themselves are

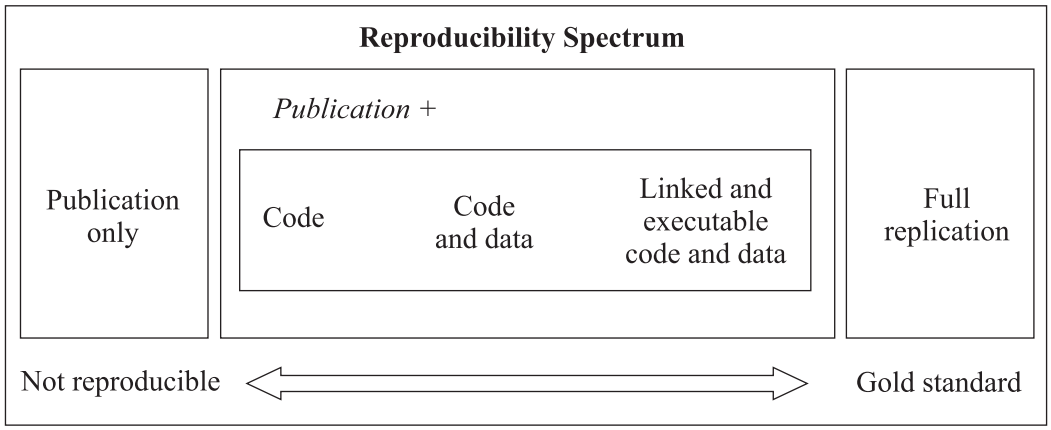


Figure 1.2: The reproducibility spectrum shows the range of how reproducible a study can be. The figure was originally published in an article by R. Peng [16].

inherently random at a fundamental level, and even if, somehow, the exact same experiments are built elsewhere, it would be impossible to obtain exactly identical measurements. Because of this uniqueness, the HEP measurements are invaluable for preservation.

According to a recent survey that gathered responses from 1,576 researchers coming from all scientific backgrounds [19], reproducibility in science is a troublesome issue. The numbers show that more than 70% of researchers have tried and failed to reproduce another scientist's experiments, and more than half have failed to reproduce their own experiments. This is caused by "selective reporting", "constant pressure to publish" and poor analysis methodology. In HEP, analyses are not easily reproduced due to limited access to the data, code and computational resources. I explore these challenges in the second half of this thesis.

Scientific studies are convoluted and often cannot be described only as "reproducible" or not. In his article, R. Peng [16] introduces a range of reproducibility as shown in Figure 1.2. The range was explained by the availability of data and metadata, and it essentially classifies studies as reproducible, partially reproducible and not reproducible.[1] In the first extreme are studies that follow gold standards and enable full replication, while the other extreme represents studies where only non-reproducible publications are available. V. Stodden introduces a more nuanced spectrum of reproducibility considering the availability of research resources [20, 21]:

- **Computational reproducibility** or bit-wise reproducibility is possible when all research resources that lead to a published result are available. They enable a full replication of the original result using the same experimental and computational tools.

- **Statistical reproducibility** occurs when only a selection of analysis resources is available

[1] Metadata is defined as "data that provides information about other data".

and the experiment can be reproduced with a small statistical error. This can happen when data samples are very large, and due to some missing information, it is still possible to produce a sufficiently similar result, even though they might not be exactly the same.

These definitions help us better understand and envision reproducibility in HEP (and also in the following chapters of this thesis). Often when bit-wise reproducibility is not possible, we should aim to enable statistical reproducibility of physics analyses.

We define partial reproducibility in HEP with preservation of *data analysis*, shown in Figure 1.1. This implies the preservation of the analysis software, derived input data and well-defined analysis workflow. Full reproducibility is reachable with preservation of both *data production* and *data analysis*. Such efforts include the preservation of the centralised experimental software and raw experimental data, in addition to the preservation of analysis resources. Enabling the complete bit-wise reproducibility is the goal, and throughout this thesis, we explore how it can be achieved.

1.4 The LHC timeline

The Large Hadron Collider (LHC) is currently the world's largest and most powerful particle collider and the largest single machine in the world. It lies in a circular tunnel of 27 kilometres in circumference, approximately 100 meters underground beneath the French-Swiss border near CERN. The tunnel was previously used for the LEP collider that operated from 1989 to 2000 [22]. The LHC was designed to operate with proton beams and heavy ion beams.

The LHC is home to four large particle physics experiments: ATLAS, CMS, ALICE and LHCb. All the experiments, together with the LHC share the same schedule regarding the time periods of active running, data collection and technical stops. During short technical stops that occur at the end of the year, the technology and experimental setup is maintained, typically without significant changes in the detectors. During long technical stops (called long shutdown) that can last for a number of months, many experimental upgrades can take place. Periods of active data collection, shown in the Gantt chart in Figure 1.3, are classified as follows:

1. Run 1 (2010 - 2013)
2. Run 2 (2015 - 2018)
3. Run 3 (planned 2020 - 2023)

4. Run 4 (planned 2025 - 2029)

5. Run 5 (planned 2030 - 2033)

The first data-taking, also known as Run 1, took place from 30 March 2010 to 13 February 2013. The primary physics programme involving proton-proton (pp) collisions was recorded at the initial centre-of-mass energy of 7 TeV (3.5 TeV for each beam). Other smaller data collection programmes were conducted at lower energies during special LHC runs. For instance, the LHCb experiment mostly studies pp collisions, but it can also study collisions of proton-helium, proton-neon etc. In 2012, the energy in pp interactions was increased to 8 TeV, and the LHC was the holder of the world record for the highest-energy particle collider at that time [23].

At the beginning of 2013, the LHC operations were paused until the summer of 2015 (Long Shutdown 1). The operations were resumed for the second data-taking period, Run 2, with pp collisions at the increased energy of 13 TeV (6.5 TeV per proton beam).

The LHCb experiment has published many groundbreaking results since its creation, such as the observation of direct CP violation in B decays in 2013 [24] and the observation of an exotic pentaquark particle in 2015 [25]. The ATLAS and CMS experiments have observed a particle consistent with the Higgs boson in 2012 [8, 13], which led to Peter Higgs and François Englert, the theorists who predicted the particle, being awarded the Nobel Prize in Physics in 2013.

1.5 Challenges

As discussed in the previous section, the LHC experiments have a lifetime of several decades, during which they constantly evolve by implementing regular detector upgrades. The time scale is one of the major challenges in scientific preservation as the detector upgrades result in changes in both data formats and software over time, thus introducing incompatibility issues in transparently accessing and analysing both old and new experimental data.

In the following sections, we explore this and other challenges of scientific preservation and reproducibility in HEP. The main challenge lies primarily in storage and handling of the vast amounts of data produced at the LHC. In addition, physics analyses are often convoluted and unique, and they also need to be captured and preserved for the future.

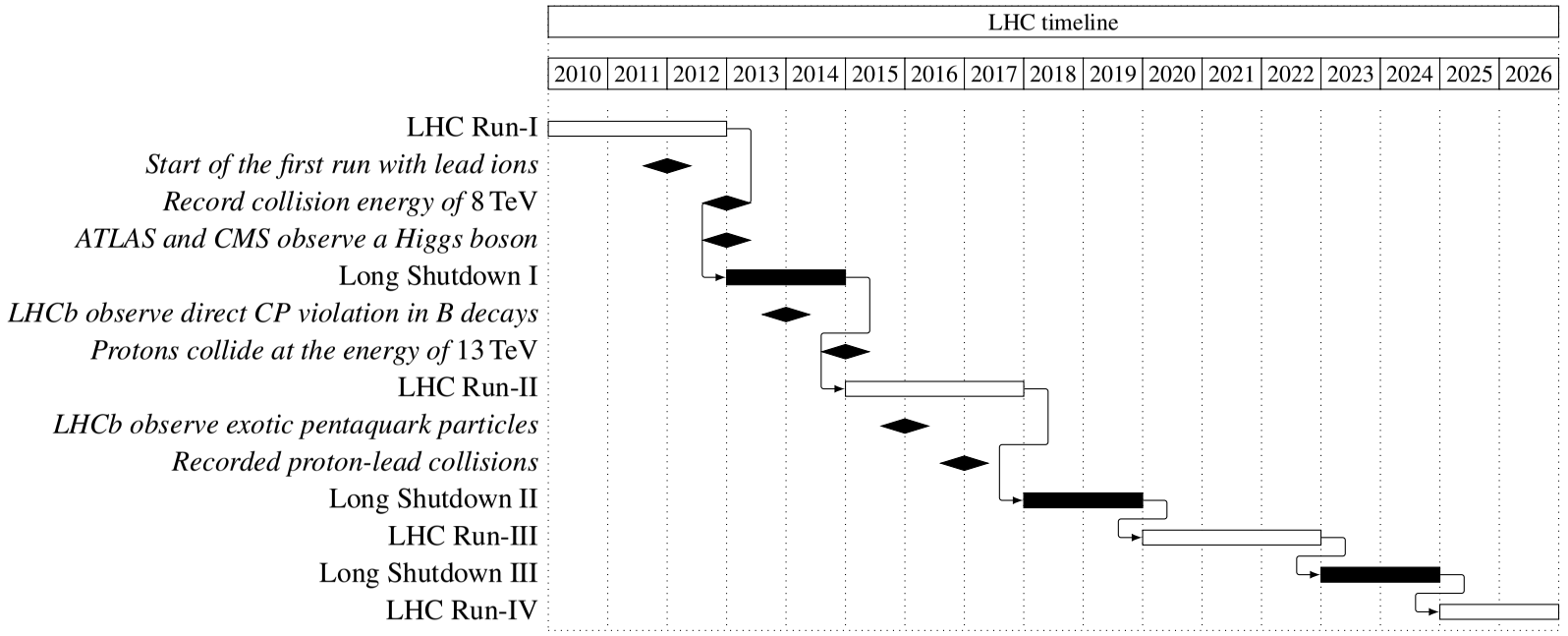


Figure 1.3: Gantt chart of the LHC timeline and a selection of CERN milestones.

1.5.1 CERN's big data

Big data is a novel industry phrase to define extremely large datasets that may only be computationally handled. The big data can be structured and unstructured, and it is often analysed using machine learning techniques. The physics data collected by the experiments at CERN certainly qualifies as big data.

In June 2017, the CERN data centre passed the milestone of 200 PB of data permanently archived in its tape libraries [26]. In 2016 alone, the raw physics data collected across the LHC experiments sums up to 50 PB. Putting it in perspective with the industry leaders: Google searches sum up to 98 PB per year and Facebook uploads to approximately 180 PB per year [27]. The LHCb experiment, which is the primary focus of this thesis, collected 9.89 PB of raw experimental data in 2016. The data volume at LHCb is expected to increase by an order of magnitude in the next run of the LHC, which is further discussed in Chapter 2. Undoubtedly, data growth is following an unstoppable trend, set to continue in the following years. As scientific research is performed both by industry and by academia, the fields can learn from each other to identify the best practices in data analysis and preservation, which we further explore in Chapter 8.

The experimental data needs to be preserved in several copies stored at different locations for data redundancy reasons.[2] Furthermore, specialised software and documentation need to be preserved in order to understand and process the data. Due to its large volume, the processing requires extensive computational resources, which are often unavailable for external researchers who aspire to reproduce HEP research. We explore possible alternatives in Chapter 5.

1.5.2 Scientific Software

There is specialised software for data production created and maintained by the experiment collaborations. In the last 15 years, more than 20 million lines of program code have been written by the analysts and computing professionals across the four LHC experiments [29]. This code is created to manage the entire data flow from data collection to final data analyses, and it is often specific for each experiment.

The challenge of preserving this software is in capturing its runtime environment dependencies and linking each of the software versions with the compatible experimental datasets. For example, an application can be tailored to work only with the experimental data of 2016 and

[2] Data redundancy is a condition created within a database or data storage technology in which a dataset is copied and held in separate places [28]. This is typically done for backup and recovery purposes.

nothing else.

On the other side, physicists doing analyses, *analysts*, develop software to implement their analysis methods. This software is almost always tied to particular datasets that were used in the analysis. Likewise, to preserve this software, it needs to be well-documented and captured with the external dependencies.

The runtime environment consisting of the hardware, operating system, compilers, and other system and software libraries change over time. This means that every time an analyst upgrades their operating system or buys a new computer, the probability of successfully preserving or later reproducing an old study decreases. When advocating for bit-wise reproducibility, it is important to note that research results often depend on seemingly small configurations in the computational environment. For example, running the same code on a 32-bit computer architecture may produce different results than running it on a 64-bit one, even if the rest of the environment is kept identical. Nevertheless, the majority of the HEP experimental code was created for a single computer architecture [29].[3] The preservation of the computational environment is a particular challenge which needs to be considered for the preservation efforts [31], hence we explore possible solutions in Chapter 8.

Finally, another challenge is imposed due to the fact that many of the original code developers have left the collaborations, which in practice means that some of the code is left with incomplete documentation, insufficient tests or its maintenance has reduced.

1.5.3 Physics analysis

Physics analyses are intrinsically complex, as they often consist of a number of stages of data processing, many of them dependent on each other. Analysts are typically studying more than one particle decay, and they are analysing the datasets collected under different detector conditions. This is why, data analyses are only comprehensible if they are provided with derived input data, documentation on analysis methods and its software.

Although the analysts may make a conscious effort to document all the details of their methods, that is often not enough for full reproducibility. Due to the complexity of the analyses and their dependence on the computing resources, it is difficult to note all of the details at all times. Things that seem obvious at the time are later forgotten. Therefore, the challenge of analysis preservation lies in the preservation of the data, documentation, software and runtime

[3] The standard hardware architecture used at LHCb is the x86 architecture, currently provided by AMD and Intel [30].

dependencies (defined in the previous section). We explore how specialised computing tools for tracking dependencies and recording workflows facilitate capturing physics analysis in Chapter 8.

1.6 Data preservation initiative

The initiative to preserve experimental data and physics analyses is a concept that the LHC collaborations had not made a priority. However, because analyses are done by individual physicists, it was recognised that if a preservation schema does not emerge, much of the implicit knowledge about analysis methods will be lost. This is why, in the recent years, the collaborations have supported several groups and task forces that make an effort towards scientific preservation. Currently, every LHC experiment has a representative to work on not only preservation efforts, but also on open data and outreach. Here, I outline the primary drivers of this initiative.

1.6.1 DPHEP 2020 Vision

Data preservation in high-energy physics (DPHEP) [32] is a study group initiated by the HEP experiments to create a shared vision and community for supporting data preservation and long-term analysis. Their objectives are to review and document the agenda of data curation, exchange information about the analysis models and address the current status of data and software preservation. Ultimately, the objective is to find a universal standard to constitute the basis for future collaborations. In 2013, the study group was endorsed by the International Committee for Future Accelerators [33].

DPHEP have published two status reports in 2009 [34] and 2012 [32], and their vision was first presented in February 2013. The vision (*DPHEP 2020*) consists of the following goals:

1. By 2020 all archived data should be easily accessible and usable by the designated communities with clear (open) access policies.
2. Tools and services for data manipulation should be provided and thoroughly documented. They should be built in common with other disciplines and based on standards.
3. Clear targets and metrics to measure the progress should be agreed between funding agencies, service providers and the experimental collaborations. The

> data and tools should be available and accessible online through a DPHEP portal.

The group introduced critical metrics for data sharing to include the completeness and quality of open data for educational outreach, reproducibility of physics results and maintenance of the full potential of data for future discovery and reuse [35]. They are set to identify and promote digital library tools and services for open data, as well as access to HEP sustainable software.

The study group encourages data sharing and advocates that open data is the best approach for long-term data preservation and reuse. Their data management plan states that datasets and software need to be assigned a reference and descriptions and that they must be preserved using standards and metadata. This should be done by assigning unique persistent identifiers, such as Digital Object Identifiers (DOI), which are standardised by the International Organisation for Standardisation [36]. Such referencing would provide direct access from publications to input data and software.

1.6.2 Open science and CERN open data policy

Open science is the movement to make scientific research, data and papers open and accessible to the general public. CERN has been leading the way with its open data, public involvement and policies stating that all papers on physics results should be published open access. Starting from August 2017, this policy expanded to instruments [37], meaning that the acquired knowledge on physics instruments, such as particle detectors, should also be published open access.

CERN experiments are some of the largest particle physics experiments, and they consume much of the entire world's budget for high-energy physics. Because of this, CERN collaborations are committed to contributing to society. They have approved open data policies, which differ in detail, but are broadly similar to each other. These policies state that the data is restricted during an embargo period measured in years, after which it is gradually released. The length of the embargo period varies between the experiments [38–40]. At the LHCb experiment, it lasts for five years after data recording, after which 50% of the data is to be made available to the general public. The remaining 50% of the data yield is to be released after ten years [38].

Open access to the data is to be implemented gradually in four levels of increasing complexity:

1. Published results. All scientific output is always published in open access journals, with preliminary results made available in Conference Reports. Data associated with the publications will also be made available.

2. Outreach and Education. CERN experiments are committed to participating in outreach activities and education.

3. Reconstructed data. When resources are identified, the experiments will provide open access to some reconstructed level data on disk at CERN.

4. Raw data. Due to the large scale and complexity of the raw HEP data, and extensive computing resources required for reconstruction, the experiments will not devote any resources to developing any service to access the full raw dataset for the public.

Regarding data preservation, the LHCb open data policy states as follows:

Data preservation is fundamentally important for the collaboration itself, regardless of any external requirements. This is to enable collaboration members to access data for many years after it was taken and requires a consistent set of the data, associated software, metadata and conditions and documentation to be preserved.

Even though scientists are often reluctant to preserve and share their research data, it has been confirmed many times that open sharing is beneficial for science and society. A notable example is the unprecedented progress made on the mapping of the human genetic code (DNA sequence) through the open data repository GENBANK [7].

Open data encourages *citizen science*, as both amateurs and professionals can access the data. Citizen science represents scientific research conducted by members of the general public, typically as part of a collaborative project with professional scientists. It can involve collection or analysis of research data, which can lead towards education or even scientific publications. An example of exceptionally successful citizen science project occurred in astronomy with the open data by the Sloan Digital Sky Survey (SDSS) project. It encouraged a significant endeavour in citizen science to produce more than 50 scientific papers [7].

Open access and open data policies are powerful steps toward open science. In the following sections, I introduce the CERN Open Data portal and give examples of cases where CERN's open data has allowed significant scientific progress to be made.

1.6.3 Funding agency requirements

There is an increasing number of policies introduced by the leading scientific journals and government institutions, which require of authors to create a management plan to preserve thier research data and share it publicly or on demand. This is imposed with a purpose to allow reproducibility and promote reuse and further research. I introduce and quote a few of these policies, which are directly linked to the high-energy physics funding and publishing agencies.

In 2014, the journal *Nature* introduced a condition of publication, which requires of the authors *"to make materials, data, code, and associated protocols promptly available to readers without undue qualifications"* [41].

The National Science Foundation (NSF) is a United States funding agency that supports fundamental research and education in science and engineering. NSF states on their data sharing policy [42]:

> *Investigators are expected to share with other researchers, at no more than incremental cost and within a reasonable time, the primary data, samples, physical collections and other supporting materials created or gathered in the course of work under NSF grants.*

Additionally, on data management plan requirements they state:

> *Proposals submitted or due on or after January 18, 2011, must include a supplementary document of no more than two pages labelled "Data Management Plan". This supplementary document should describe how the proposal will conform to NSF policy on the dissemination and sharing of research results.*

The Science & Technology Facilities Council (STFC) is one of the leading funding bodies in the UK, which supports research in particle physics, nuclear physics, astronomy and other sciences. They introduced a scientific data policy in 2009, to encourage that the data produced under the STFC funding should be "carefully managed and optimally exploited, both in the short and the long term" [43].

> *The key principle of the [scientific data] policy is that all funded activities are required to have a data management plan, which must be in line with recommended good practice. These individual plans will then have the added check of being subject to approval by the relevant STFC boards and panels.*

Many of the policies require a plan for data management and preservation. However, it is important to note that there is no recognised standard for data analysis and preservation, thus this is often something left to the researchers to arrange. We explore reproducibility and reusability in the current analysis documentation in Chapter 7 and introduce a standard for analysis preservation in Chapter 8.

1.7 Related work

There are several HEP projects tackling the challenges of data preservation and reproducibility. The most notable ones are the CERN Open Data portal [44], the CERN Analysis Preservation portal [44], the HEPdata portal [45], Everware [46] and Swan [47].

1.7.1 CERN Open Data portal

The CERN Open Data (COD) portal [44], based on the Invenio Digital Library [48] framework, is an access point to data produced by the experiments conducted at CERN. In addition to the data, the portal provides access to free and open source software to read and analyse the data and the corresponding documentation.

Creating a new chapter in HEP history, the first LHC experiment to release their data was CMS. They published approximately 20 TB of collision data from the first LHC run in November 2014 [49]. Due to its complex format and large volume, it is an ongoing discussion whether this data can find its use outside the HEP collaborations. However, proving that the LHC data analysis outside the CERN infrastructure is indeed possible, the data was used for scientific research in HEP theory already in 2015. The research was conducted by Professor Jesse Thaler and a team of theoretical physicists who analysed the data to learn about the substructure of jets [50]. They have released a document [51] where they describe their experience with the CMS Open Data as "fantastic" and give their feedback and suggestions to the further development on open data.

The portal includes an educational section with simplified datasets and tools for outreach and training. Such resources were provided by the ALICE, ATLAS, CMS and LHCb collaborations, in addition to physics exercises by the International Masterclass organisation [52]. The target audience for this section is high school and university students.

There were other smaller efforts to release HEP data. In particular, the ATLAS experiment has released their simulated data in the form of an online data analysis challenge called

the "Higgs boson machine learning challenge" hosted on the Kaggle platform [53]. Following their success, the LHCb experiment organised a challenge called "Flavours of Physics: Finding $\tau \to \mu\mu\mu$" on the same platform [54]. Competitors were given the real and simulated data to analyse, thus encouraging LHCb research outside of the collaboration. This data is to be available as open data on the COD portal.

1.7.2 CERN Analysis Preservation

CERN Analysis Preservation (CAP) [44] is a web platform for preserving knowledge and assets of an individual physics analysis. It implements a flexible approach, which can conform to a wide range of HEP analyses. The platform can preserve everything from presentations and publications to experimental configurations and large volumes of data. It is available through both a user interface on a web browser and a command-line client, which are in detail explored in Chapter 8.

Special attention has been taken to comply to the collaboration policies on access permissions of the analyses. In particular, whilst the measurements are being made in collaboration, they are typically kept private. Once a result has been verified in a series of reviews, it is published, sometimes together with other analysis resources. This is mostly because the collaboration does not want to disclose preliminary results that have not been properly verified. The CAP service conforms to and implements the collaboration access restrictions. Analyses on CAP can be private, internal or public.

The portal provides advanced search for analysis information, and ultimately it should serve as a search engine for HEP analyses. It is developed as a collaboration across the LHC experiments, the CERN Information Technology (IT) department and the CERN Scientific Information Service (SIS) group.

1.7.3 HEPdata

The Durham High-Energy Physics Database HEPdata [45, 55] is an open access repository that gathers the experimental results from HEP publications. These results may be presented in the form of plots, tables and final (derived) datasets, which can be explored on a web page in an interactive and accessible way (shown in Figure 1.4). Currently, the portal gathers results from several thousand publications, and it allows browsing, searching and studying of the plots and tables linked to the publications.

Although HEPdata shares the results on physics analyses, it does not provide the research

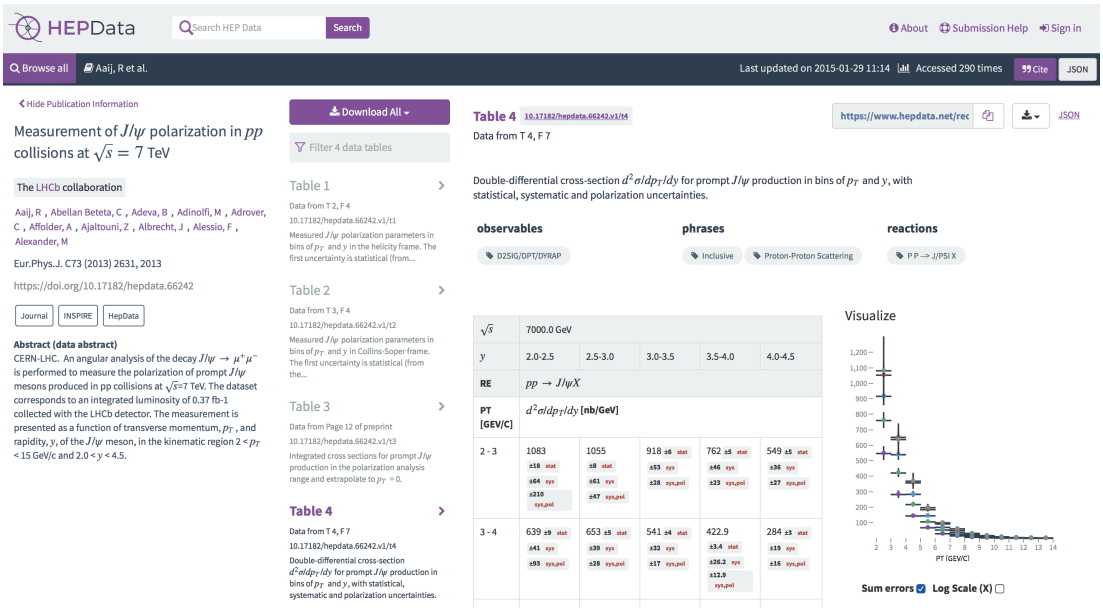


Figure 1.4: Appearance of the HEP data portal in a web browser [56].

methods on how the results were obtained. This is why an approach for preserving and sharing analysis methods and workflows needs to be identified.

1.7.4 Everware and Swan

Everware [46] is an open-source project created to help in analysis preservation, reproducibility and educational activities. It allows performing physics analyses in Jupyter notebooks [57], which are integrated into a web browser. These notebooks contain a mixture of the source code, comments and output that can be plain text, tables, figures or animations. The computational environment of the notebooks contains the libraries required for the analysis code to run. Everware facilitates sharing results and collaborative work, as the user interacting with the notebook can study and execute the code to get always "the same" output. Currently, Everware is actively used for educational purposes at workshops and summer schools, where it allows the participants to start on the same page. However, its use in the HEP analysis development is still uncertain.

A similar project developed by the CERN EP-SFT (Experimental Physics Software) group is called Swan [47] (short for "service for web-based analysis"). This project resembles Everware as it makes use of the Jupyter notebooks to provide a platform for interactive data analysis on the cloud. It allows writing and executing analysis code on a web browser as shown in Figure 1.5. Swan provides access to the official CERN software and the storage system, which is essential for HEP data handling. Hence, Swan is well-integrated into the CERN infrastructure

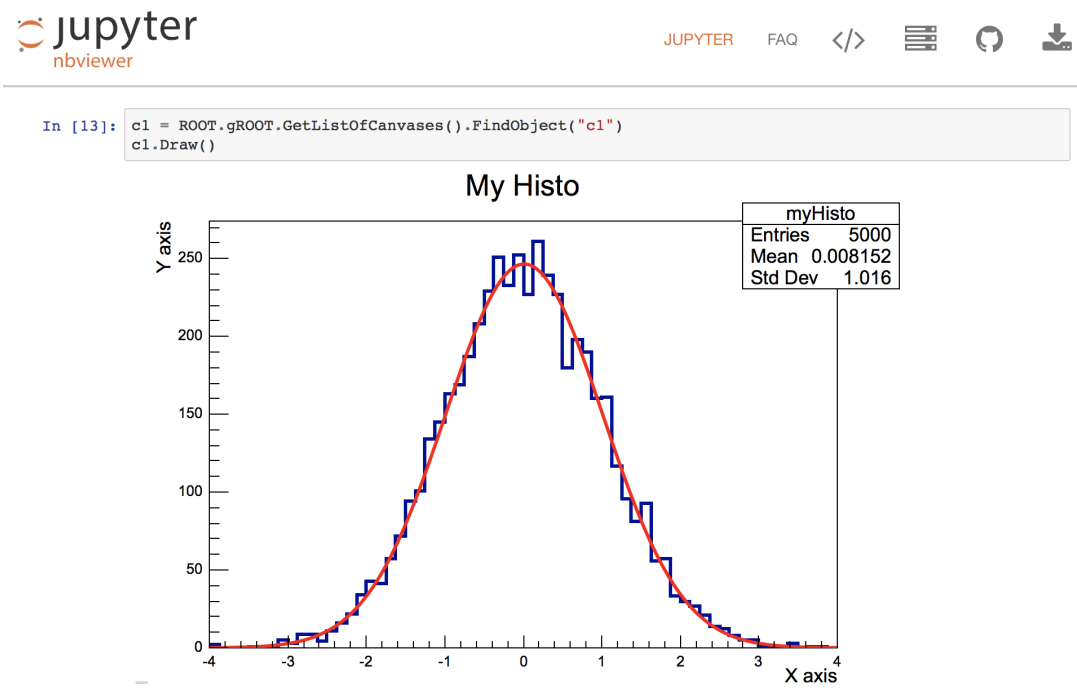


Figure 1.5: Swan appearance in a web browser [58].

and it enables a convenient way to demonstrate and share results, code and analysis methods.

Both Swan and Everware received positive feedback when they were introduced at CERN workshops. However, there were concerns on "dependency handling", as each step in data analyses is often dependent on others. The projects are said to be useful for "high-level analysis steps", which presumably means presenting analyses in their final stage when the datasets are relatively small. Nonetheless, Swan can be linked to strong data processing engines, eliminating the constraints on processing high data volumes. In addition, it is continuously maintained and improved.

1.8 Chapter summary

In this chapter, I introduced an example of the HEP measurement lifecycle and the motivation to preserve the experimental data and analyses. There are many challenges in the field, which lead to an emergence of a number of new projects to address them.

The CERN Open Data portal is created to promote open science by sharing data and educational resources. The CERN Analysis Preservation portal captures analysis workflows and methods, which is something that is not commonly captured in physics publications. HEPdata collects and shares the physics results, and Everware and Swan facilitate analysis presentation and reproducibility.

Many of these projects are new, and they have tremendous potential in analysis preservation, reproducibility and result sharing. In the long run, they can improve how research is done and speed up the rate of scientific discovery. However, adopting and employing new tools is not always easy, and these challenges are discussed later in the thesis.

To advance the preservation initiative, an effort has to be made within each collaboration. In the next chapter, I introduce the LHCb experiment, followed by ideas and solutions to various problems in data preservation.

Chapter 2

The LHCb experiment

The LHCb experiment [59] is one of the particle physics experiments at CERN. In this chapter, I introduce the LHCb physics programme and the detector. I explain the process of data collection and discuss the volume of data collected so far at LHCb. Finally, I talk about the experimental upgrade and how we expect to deal with the increased data rate in Run 3 of the LHC.

2.1 The Large Hadron Collider

The LHC, introduced in Section 1.4, is currently the most powerful particle collider in the world. The LHC complex accelerates proton beams in a succession of machines with increasingly higher energies, as shown in Figure 2.1. First, the protons are taken from hydrogen atoms and injected into the linear accelerator LINAC 2, where they reach the energy of 50 MeV. Then they travel through the PROTON SYNCHROTRON BOOSTER, PROTON SYNCHROTRON (PS) and SUPER PROTON SYNCHROTRON (SPS), reaching the energy of 450 GeV before they are injected into the LHC to reach the maximum energy of up to 6.5 TeV [60] per beam.

The LHC is comprised of two beam pipes in which particles circulate in opposite directions. The LHC radio-frequency cavities [61] are used to accelerate the particle beams. Due to the oscillations of the voltage inside the cavity, the particle beam is sorted into discrete packets called "bunches". Dipole magnets are used to bend the trajectories of particles to guide them about bends of circular accelerators [62]. There are more than 1,200 main dipoles in the LHC, each 15 m long and weighing approximately 35 t [63]. The particles need to be focused together to increase the probability of their collisions inside the LHC detectors. For this purpose,

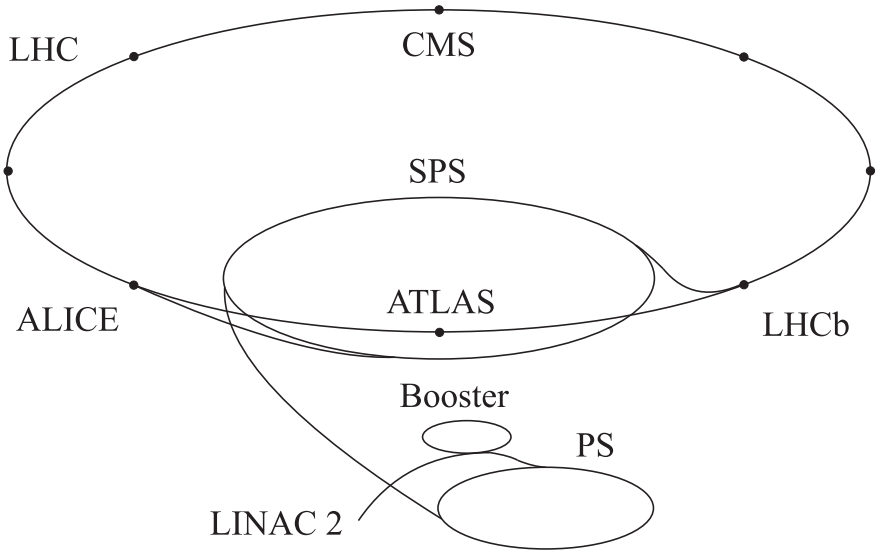


Figure 2.1: The CERN accelerator complex

quadrupole magnets are used, which have four magnetic poles arranged symmetrically around the beam pipe to focus the beam either vertically or horizontally. The LHC cooling system keeps the magnets and radio-frequency cavities working at the temperature close to absolute zero, allowing them to work in a superconducting state without losing energy to resistance.

The beam pipes cross at several points along the LHC perimeter allowing the particle bunches to collide. These points are where the LHC experiments are located. The four largest experiments are:

ALICE (A Large Ion Collider Experiment) [64]: an experiment dedicated to heavy ion physics. It is designed to address the physics of strongly interacting matter and quark-gluon plasma at extreme energy densities and temperatures in nucleus-nucleus collisions. Quark-gluon plasma is hypothesised to have existed just after the Big Bang. In addition to protons, the LHC collides lead ions specifically for ALICE, producing temperatures more than 100,000 times hotter than the Sun. The ALICE detector is 26 m long, 16 m high and 16 m wide and it weighs about 10,000 tons.

ATLAS (A Toroidal LHC Apparatus) [10]: a multi-purpose detector. Designed to cover a wide range of physics, including the search for the (now found) Higgs boson, supersymmetry (SUSY), dark matter and extra dimensions. It is 46 m long, 25 m in diameter and it weighs about 7,000 tons. Its dimensions make it the largest particle detector ever built.

CMS (The Compact Muon Solenoid) [11]: the second general purpose experiment. It has

a different design and different technical solutions than ATLAS, though it studies the same physics phenomena. The CMS detector is 21.6 m long, 15 m in diameter, and weighs about 14,000 tonnes, which makes it the second largest (but the heaviest) experiment at the LHC.

LHCb (Large Hadron Collider beauty) [59]: an experiment that has a wide range of physics programmes, both as a heavy flavour experiment focused on the matter-antimatter asymmetry, and as a general purpose detector in the forward region.[1] It is the smallest of the four detectors, with a length of 21 m, a height of 11 m, a width of 13 m and weight of about 5,600 tonnes.

2.2 LHCb physics programme

The original LHCb physics programme was focused on testing the Standard Model of particle physics and on searching for New Physics effects that go beyond the scope of the Standard Model. These tests are conducted by studying CP asymmetries and rare decays of b-hadrons produced in pp collisions in the LHC. Since the first run of the LHC, the physics programme has vastly expanded to include, for instance, topics like electroweak physics and exotics searches [65], which previously were not a part of the programme.

2.2.1 The Standard Model

The Standard Model of particle physics is a theory that describes elementary particles and their interactions. The particles are grouped into a set of six quarks and six leptons, in addition to the Higgs and force-carrying bosons (as shown in Table 2.1). The model has been remarkably successful at predicting the behaviour of elementary particles and has been rigorously tested in many particle colliders. However, there are several experimental observations that cannot be explained by the Standard Model. In particular, it does not explain the amount of visible asymmetry between matter and antimatter in the universe.

For every particle in the Standard Model, there is an antiparticle, which, like matter, can combine to create anti-atoms and anti-molecules. Antiparticles have the same mass and spin as their corresponding matter particles, but their quantum numbers are inverted. When matter meets antimatter, they annihilate and convert into other particles. Some particles (gluon,

[1] Forward region means that LHCb, as a single arm spectrometer, covers small angle with respect to the beam line, in the pseudo-rapidity region $2.0 < \eta < 5.0$.

	Fermion generations			Gauge bosons	Scalar bosons
	I	**II**	**III**		
Quarks	u up	c charm	t top	g gluons	H Higgs
	d down	s strange	b bottom	γ photon	
Leptons	e electron	μ muon	τ tau	Z^0 Z boson	
	ν_e electron neutrino	ν_μ muon neutrino	ν_τ tau neutrino	$W^\pm$ W bosons	

Table 2.1: Elementary particles in the Standard Model of particle physics.

photon, Z, and Higgs) do not have a separate antiparticle. These can be thought of as "their own antiparticle".

There are four known fundamental forces between particles. The interactions are explained as the exchange of fundamental particles called the force carriers. These particles are bosons, and each force is associated with a different boson. The carrier of the electromagnetic interaction is the photon (γ). It is a quanta of light and it has no charge. The carriers of the strong force are called gluons (g). There are eight types of gluons, each having a different strong charge. Gluons have not been observed directly, as they cannot be produced as free particles. The weak interaction is characterised in two types: the one that carries electric charge via $W^\pm$ bosons and the one which is electrically neutral and transmitted via Z^0 bosons. Lastly, there is gravity with the graviton as its fundamental particle. The Standard Model does not include the theory of gravity.

2.2.2 CP violation and matter asymmetry

The laws describing conservation of energy, momentum and angular momentum are exact consequences of the symmetries of space and time. Besides, symmetries lead to conservation of various quantum numbers, which are labels of a physical state. Some examples of quantum numbers are the electric charge, the spin of a particle, baryon number etc. Conservation laws put strong constraints on which processes can occur in nature.

The charge-parity (CP) operation in a particle physics theory is the product of charge conjugation (C) and parity (P) operations. Charge conjugation reverses all the internal quantum numbers of the particle transforming it into its antiparticle. Parity creates a mirror image of a physical system (as shown in Figure 2.2). If P-symmetry is conserved, then the mirror image reaction occurs at the same rate as the original reaction. However, this is proven to be false, as weak interactions violate P-symmetry [66]. In 1957, Lev Landau proposed that CP-symmetry is the true symmetry between matter and antimatter [66]. This means that a process where all particles are exchanged with their antiparticles was assumed to be equivalent to the mirror image of the original process. The electromagnetic and strong interactions seem to be symmetric under the combined CP transformation operations, but again this symmetry is slightly violated in weak interactions. This violation was observed in 1964 in the decays of neutral kaons [67]. James Cronin and Val Fitch were awarded the Nobel Prize in Physics in 1980 for this discovery. In the last decades, several studies of CP violation were performed, and the measurements of direct CP violation in B_s^0 mesons were reported [68].

However, the HEP theorists included a time symmetry (T) to expand the theory, meaning that the equations of motion (or more general the action) are invariant under time reversal. It is believed that CPT-symmetry is the fundamental property of quantum field theory and that it has to be conserved unless the entire mathematical foundation of the Standard Model is wrong.

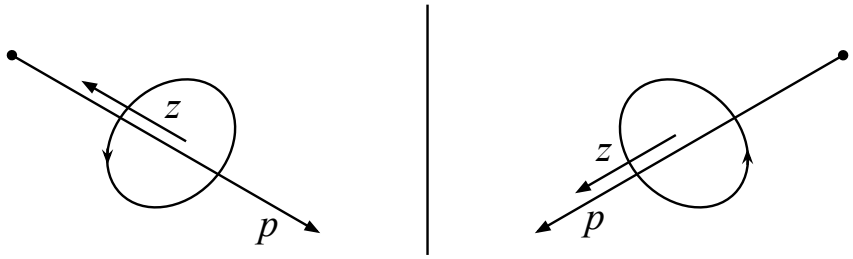


Figure 2.2: The action of *P*-symmetry on a particle with momentum p and spin z. The right image is how the system looks after applying the *P* operation.

It is believed that in the early universe, an equal amount of matter and antimatter was produced [69]. However, today's universe is composed of ordinary matter, and antiparticles are only observed in particle accelerators and cosmic ray collisions in the atmosphere. If matter and antimatter were created and destroyed together, the universe should contain nothing but leftover energy. However, a tiny portion of matter managed to "survive" and to compose the universe we see today. This means that at some point in time, the balance was broken and the currently observed asymmetry was produced because of the imperfect annihilation. The Standard Model

does predict a small asymmetry between matter and antimatter. However, the prediction is not enough to fully explain the quantity of matter we observe [70].

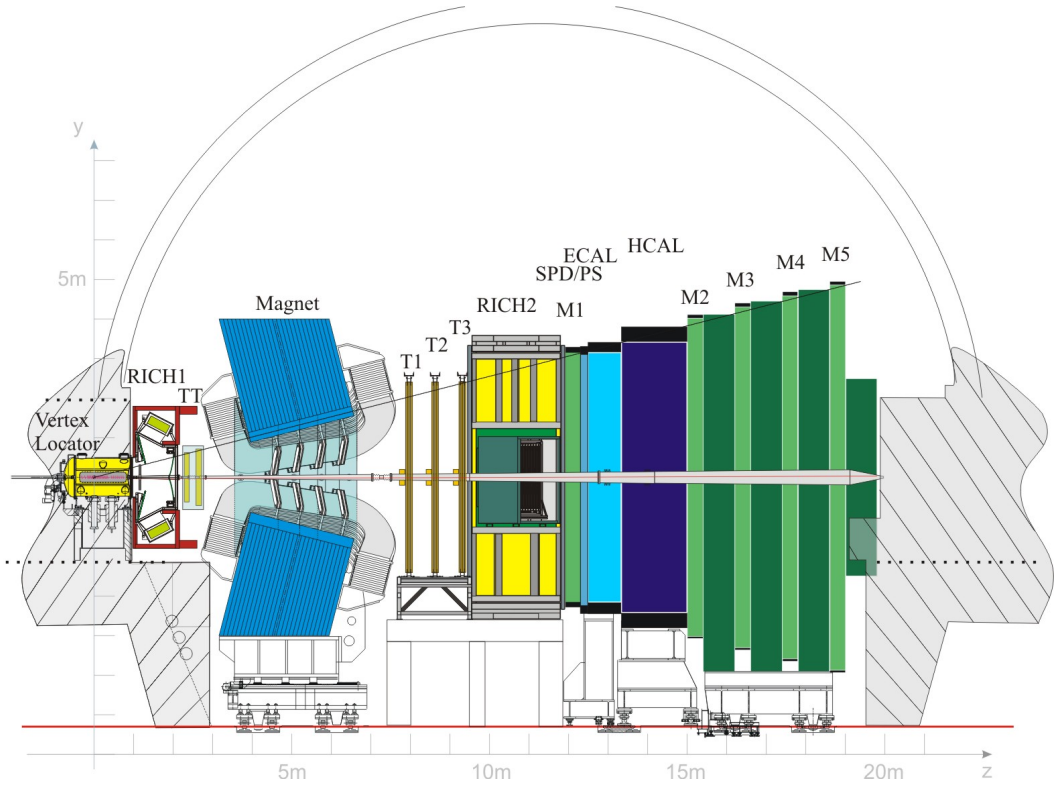
There are three conditions, called the Sakharov conditions [71] that are necessary for this asymmetry to arise. The first condition states that baryon number conservation was violated in the early universe. Baryons are one of the major group of particles, which includes protons, neutrons and many more massive particles with similar properties. According to the theory, the number of baryons was higher than the number of anti-baryons in the early universe, which led to a large number of baryons today and the lack of anti-baryons. Secondly, there must be C and CP violation, because if there is not, every reaction which creates a net number of baryons over anti-baryons would be negated with a CP conjugate reaction generating the opposite. The last condition states that a departure from thermal equilibrium must have occurred since any baryon-number violating process would have been balanced out by the inverse reaction in equilibrium. Essentially, there was a point in time when these processes froze out to create the imbalance. It persisted thereafter because the processes that could reverse it were no longer possible.

2.3 The LHCb detector

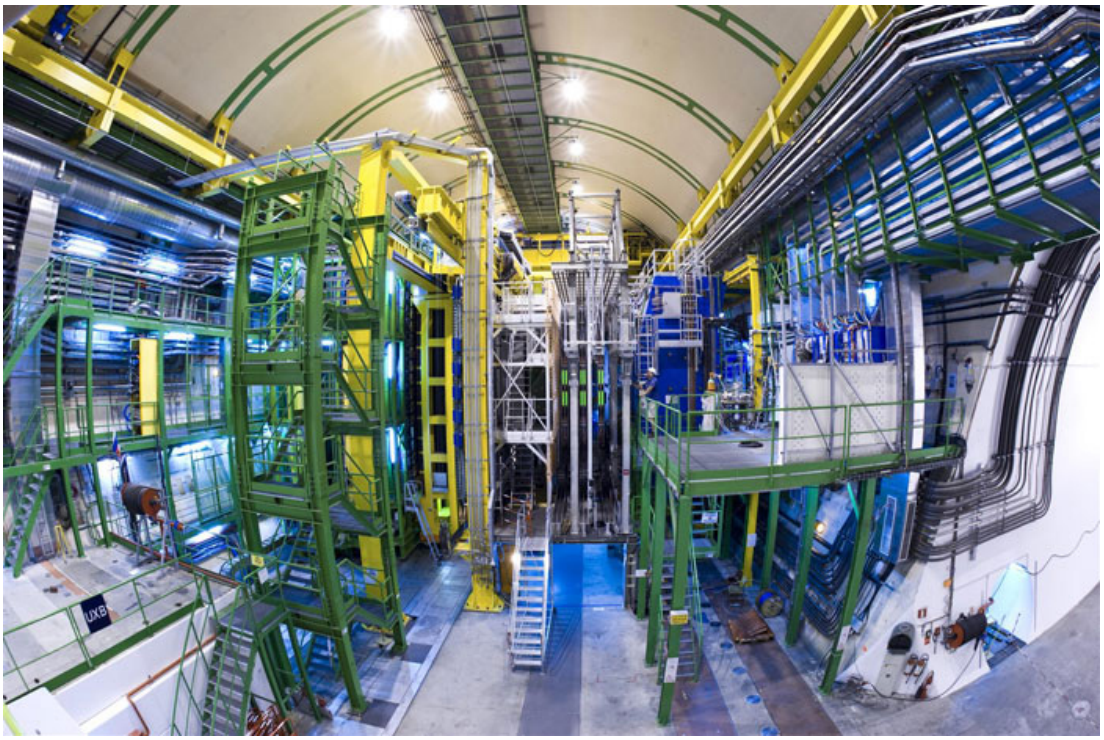
The LHCb collaboration designed a unique forward spectrometer, which was originally meant to study the decays of the B-physics and make precise measurements of the CP violation phenomenon. Today, LHCb is a general purpose detector in the forward region as shown in Figure 2.3a and Figure 2.3b. The LHCb coordinate system is a right-handed Cartesian system with the origin in the interaction point. The x-axis is oriented horizontally towards the outside of the LHC ring; the y-axis is pointing upwards with respect to the beam-line and z-axis is aligned with the beam direction. Its experimental setup consists of the Vertex Locator (VELO), tracking stations, calorimeters, Cherenkov detectors (RICH 1 and 2) and the muon system. The components are roughly grouped into two systems with separate tasks: tracking and particle identification (PID).

2.3.1 The tracking system

The tracking system comprises of the VELO surrounding the pp interaction region, a silicon-strip detector (TT) located upstream of a dipole magnet, the magnet, three stations (T1–T3) of silicon-strip detectors and straw drift tubes placed downstream of the magnet [72].

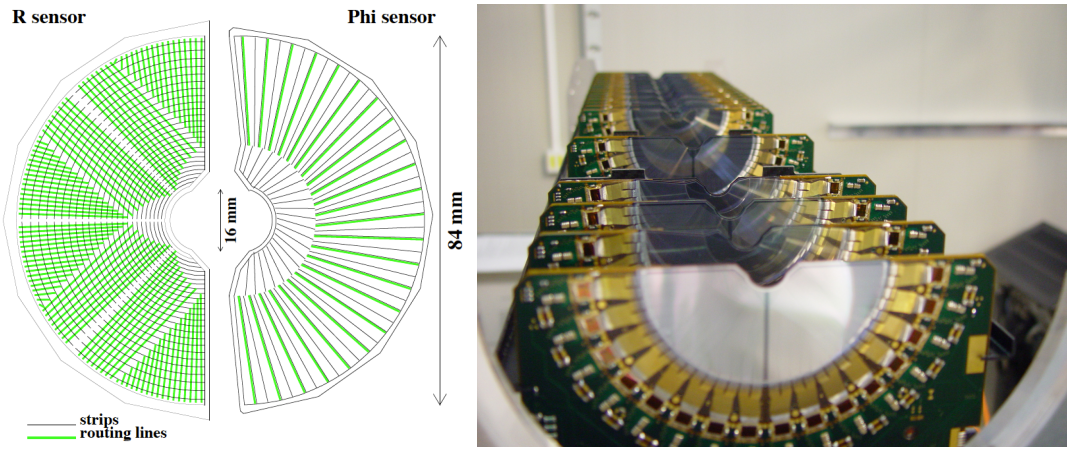


(a) The schema of LHCb detector.



(b) A photograph of the LHCb experiment inside the cavern. On the left side are the muon detectors and calorimeters. Roughly at the centre is the RICH and the magnet. The collision point and the VELO are located on the right side.

Figure 2.3: The schema (a) and a photograph (b) of the LHCb experiment.



(a) VELO schema with the routing lines oriented perpendicular and parallel to the silicon strips. Routing lines are conduction lines carrying a signal current from the strips to the edge of the sensor.

(b) Photos of VELO during assembly.

Figure 2.4: Schema and photo of the Vertex Locator sensors.

The Vertex Locator

VELO is a silicon detector that provides precise measurements of track coordinates close to the interaction region. It is composed of 84 single-sided radial (R) and axial-angle (ϕ) measuring strip sensors (shown in Figure 2.4). The VELO operates in a secondary vacuum tank inside the LHC beam pipe [73] in order to minimise the material traversed by the particles and to allow the sensors to be as close as possible to the beam line. The VELO consists of 42 modules mounted perpendicular to the beam, with each module consisting of one *R*- and one ϕ-sensor. The modules are distributed over a length of 1 m along the beam.

During the LHC data-taking period, the VELO is exposed to high radiation doses. To preserve the detector, the VELO has two movable halves. During beam injection, the halves are kept at a distance of 3 cm away from the operating position, while during the data-taking, when the beam is calibrated and stable, the innermost part of the halves is kept at a distance of 8 mm from the beam. These positions are called *VELO open* and *VELO closed* respectively.

Tracking stations

Tracking is the process of reconstructing the trajectories of charged particles. In addition to the VELO, there are four tracking stations at LHCb. They are Tracker Turicensis (TT) and the three T stations (T1–T3), which are separated by the magnet (as shown in Figure 2.3a). These tracking stations provide measurements of momentum and charge of charged particles in the

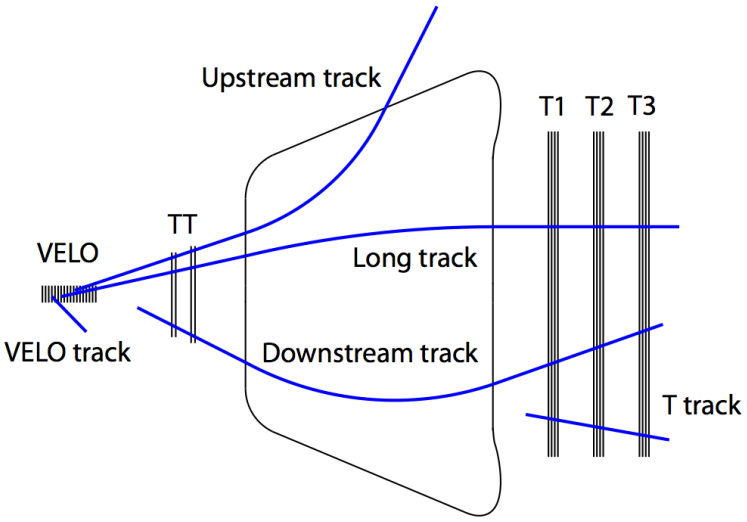


Figure 2.5: A schema of various particle tracks traversing through the LHCb detector.

magnetic field.

The T1–T3 are divided into two parts: the inner tracker (IT) and the outer tracker (OT). These two detectors were built with different technologies. The IT and the TT use silicon microstrip detectors, whereas straw tubes are used in the OT. The IT has a finer granularity than OT, and it is located in the inner region close to the beam pipe. The OT is lower in cost and occupancy, hence it is located in the outer region where high granularity is not required [74].

Track reconstruction

Track reconstruction is done using information left by charged particles traversing the tracking system. This information is recorded as particle hits in the VELO, TT, IT and OT detectors [75], which are little track segments that are combined together to form the particle trajectories. First, the VELO tracks and tracks from the T stations are considered, which are then extrapolated to form longer trajectories through the length of the LHCb detector. These tracking algorithms also infer the particle momentum.

The reconstructed tracks are classified into five types (as shown in Figure 2.5):

- Long tracks traverse the full tracking system. They hit the VELO, T1–T3 stations and the TT stations. Because they are able to traverse through the full magnetic field, they have the most precise momentum estimate and therefore are the most important set of tracks

for physics analyses.

- Upstream tracks are the ones passing through only the VELO and TT stations. Their momentum is too low to pass the magnet and reach the T stations, resulting in poor momentum resolution.

- Downstream tracks are the ones that pass only through the TT and T stations. They are produced by the decays of long-lived neutral particles, which pass the VELO before decaying.

- VELO tracks are formed by particles typically passing at a large angle or backwards. They could not be reconstructed into a longer track, and thus, they are generally used for the primary vertex reconstruction.

- T tracks are the ones that pass only through the T1–T3 stations. They are typically a product of secondary particle interactions.

2.3.2 The particle identification system

Particle identification is essential for the LHCb physics programme to distinguish between charged particles like electrons, muons, pions, kaons and protons traversing through the detector. The particle identification (PID) system is composed of the RICH 1 and 2, electromagnetic and hadronic calorimeters, and the muon stations.

The PID detectors have very different reconstruction algorithms, but all of them allow the computation of likelihood ratio between particle hypotheses for each reconstructed track [76]. The likelihood ratios of the PID detectors are combined to form the global particle identification variables, which are used as selection criteria in data analyses.

Cherenkov detectors

The ring imaging Cherenkov detectors, RICH 1 and RICH 2 [77, 78] (as shown in Figure 2.3a), are two sub-detectors of LHCb that are predominately used for hadron identification. Their role is essential in classifying charged hadrons (kaons, pions and protons) that are frequently produced in b particle decays. The most abundant particles in pp collisions are pions, hence the RICH identification algorithms start by assuming that all particles are pions. This is followed by recomputing how likely each track is to be an electron, muon, pion, kaon or proton. The

(a) View from the LHCb cavern.

(b) Downstream view of the ECAL installation (but not completely closed) with the exception of some detector elements above the beam line.

Figure 2.6: Pictures of the electromagnetic calorimeter.

combination that gives the optimal event likelihood is identified and the mass hypothesis for that track is assigned. This process is then repeated for all tracks in the event.

The RICH 1 detector is located upstream of the LHCb dipole magnet, between VELO and the TT. RICH 1 combines two radiators, silica aerogel and C_4F_{10} gas, and it identifies charged particles in the low momentum range (from approximately 2 GeV to 60 GeV). The RICH 2 detector is located downstream of the magnet, between the last tracking station (T3) and the first muon station (M1). It has a CF_4 gas radiator and it provides PID of the tracks produced at low angles in the high momentum range (from 15 GeV to 100 GeV). The weight of the RICH 1 detector is about 16 tons and RICH 2 about 30 tons.

The calorimeter system

The calorimeter system performs several functions. It provides the energy measurements of electrons, photons and hadrons, and facilitates in their identification by assigning the likelihood of electrons relative to the pion hypothesis [76]. The first event selection system, called hardware trigger (described in Section 2.4), heavily relies on the calorimeters. More than 80% of its output is decided by candidates identified by the calorimeter system [79]. It is implemented as first an electromagnetic calorimeter (ECAL) shown in Figure 2.6, followed by a hadron calorimeter (HCAL).

The muon system

The muon system [59] is composed of five stations M1 – M5, which are rectangular in shape and placed along the beam line, as shown in green in Figure 2.3a. The first station M1 is located in front of the calorimeters, and it is used to measure the transverse momentum for the hardware trigger. Stations M2 to M5 are placed after the calorimeters. They are separated by iron absorbers that are 80 cm thick to select penetrating muons.

The system is equipped with multi-wire proportional chambers (MWPC) containing the gas mixture of Ar, CO_2 and CF_4 in proportion 40:55:5 [80]. This is the content of 99% of the total gas area in the muon system. However, the inner part of the first station uses triple-GEM detectors filled with a gas mixture of Ar, CO_2 and CF_4 in proportion 45:15:40. The full system comprises 1380 chambers and covers a total area of 435 m^2.

For particle identification, the muon system performs a binary selection of a particle being a muon or not. It detects hits in the five muon stations, forms track segments and assigns a likelihood for the muon (or not-muon) hypothesis. The muon system provides information for the selection of high transverse momentum muons at the trigger level and for the offline muon identification. Muons are easy to identify because of their relatively large mass and high penetrating power, which results in a clean signal. The minimum momentum needed for a muon to cross the five stations is approximately 6 GeV. The muon stations are the last detectors of the LHCb and ideally should be only traversed by muons.

Dipole magnet

A dipole magnet located between the TT and T stations (marked in blue in Figure 2.3a and shown in Figure 2.7) facilitates measuring the momentum and electric charge of particles travelling through the LHCb detector. Inside the VELO there is a negligible magnetic field, so all the particle tracks appear as straight lines. When they pass through the magnet, the bending of their trajectories is determined by their momentum and charge. The magnet provides a magnetic field in the y direction of 4 T m integrated along 10 m.

The direction of the magnetic field is periodically reversed and approximately equal amounts of data are recorded in each magnet polarity. This is done to understand and suppress biases caused by smaller detector asymmetries. For instance, if one side of the detector is more efficient and favours particles of one charge, by inverting the magnetic field opposite particles would be prioritised. This results in balancing out the biases in the dataset, which is important in

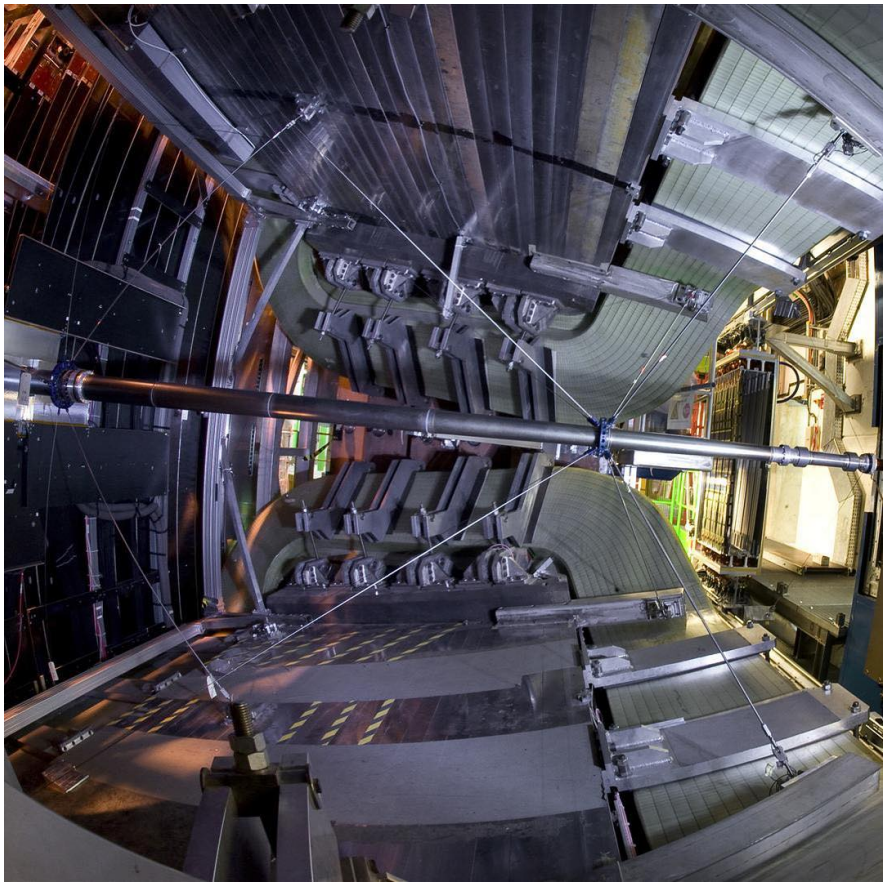


Figure 2.7: Inside the LHCb detector magnet. At the centre of the picture is the LHC beam pipe [81].

the CP measurements. In the current experimental setup, the magnet polarity can be *up* or *down*.

2.4 Data taking and the trigger system

During data collection periods, the proton bunches are injected into the LHC at 25 ns separation. Even though there are over 3,500 bunch spacings around the LHC, in practice many of them are not filled with protons. There are typically many consecutive filled bunches followed by a number of consecutive empty bunch spacings. At most about two-thirds of bunch slots are filled, however this depends on the state of the LHC and the goals of the run. The approximate frequency of bunch crossing at the LHC is 40 MHz, but the nominal ratio of proton bunch crossing is about 30 MHz, which defines the rate of data-taking.

The main challenge in data collection at LHCb is imposed by the large volume of data produced at the LHC. A *pp* collision produced in the LHCb experiment results in about 30 kB of data, meaning that a nominal data rate sums up to be approximately 1 TB per second. The

challenge lies in economically transferring and distributing this amount of data from the detector to the storage. However, most particles produced in the LHC are well-known Standard Model particles, and new exotic particles occur at a much lower rate. To select interesting particle collisions and reduce the output data rate, the LHC experiments use a mechanism called the *trigger*.

The LHCb trigger [82] is implemented in two main levels: hardware and software. The trigger implemented in hardware is known as the Level 0 (L0) trigger, and the software trigger is called the high-level trigger (HLT). The selection algorithms that either accept or reject events are called *trigger lines*. The average number of visible *pp* interactions per bunch crossing in LHCb was approximately 1.1 in 2017 (deduced from Ref. [83]), and many bunch crossings do not produce particle collisions. Therefore, the trigger is designed to identify and keep larger events (with more particle interactions) than smaller events [84].

The L0 trigger reduces the data flow from approximately 30 MHz to about 1 MHz by selecting hadrons, electrons, photons and muons. It runs synchronously to the LHC bunch crossing, and it uses only information from the calorimeters and the muon stations for its selection as they can be read out at the same rate. Muons with high momentum and energy are typical products of the decays of b and c hadrons due to their large masses, which the L0 trigger is designed to identify.

The selection is further narrowed down in the HLT, which is split into HLT1 and HLT2 to allow data analysis in real time and a reliable distribution of updated detector calibration constants in Run 2 (as shown in Figure 2.8). The HLT1 reduces the data flow from 1 MHz to approximately 150 kHz. The selected events are then temporarily saved in a buffer on disks to be further filtered by the HLT2. The HLT2 runs in the periods between individual LHC fills and during LHC technical stops and it reduces the flow to the final 12.5 kHz to be sent to storage. The reduced data flow thus allows for the use of computational resources more efficiently. The HLT is implemented in C++ and runs on the computer infrastructure called the event filter farm (EFF) [85]. The HLT performs a reconstruction of the physics properties of an event using the information from all the LHCb sub-systems. It executes a number of selection algorithms that use a set of criteria to identify exclusive or inclusive particle decays. Only events that are accepted by both the L0 trigger and HLT are stored and available for use in physics analyses.

Processes that take place in real time during data-taking, when the experiment is running, are commonly referred to as *online*. This typically includes selection in the triggers and the management of raw files. Simulation production and further processing of the raw files are done

offline, independently of the experimental data-taking.

2.4.1 The LHCb data streams

The data selected by the trigger system is organised into three main data streams as shown in Figure 2.8. They are the full stream, the turbo stream and the calibration stream [87].

The full stream was the main data streaming strategy in the Run 1 of LHC and it is used for most of the data processing in Run 2. The full stream captures and stores raw events that are later subjected to further offline processing. These events then undergo the reconstruction and preselection (also called *stripping*) steps that identify the decay channels of interest.

The turbo stream was introduced in Run 2 to process about 30% of the total event rate at LHCb. In this stream, only a subset of raw event information which is relevant for physics analyses is selected and stored. These events do not require further event reconstruction, and the size of an event is about two times smaller, therefore enabling the use of the existing CPU and storage resources more efficiently [86].

Calibration and alignment of the detector take place at the beginning of each fill. As a result of the split of the HLT in Run 2, these processes are performed in real time [88]. The automatic detector alignment procedures run at the start of each fill when a required sample of candidates has been selected, saved in the buffer and reconstructed (as presented in Figure 2.8), while the calibration constants are evaluated periodically during data-taking.

2.4.2 Instantaneous luminosity

An important measure of collider performance that is directly correlated with the volume of collected data is instantaneous luminosity. It is a measure of how many collisions occur per second inside the detector. The higher the luminosity, the more collisions happen in a second and the more particles can be produced [89]. This can be advantageous or disadvantageous depending on what physics phenomena are studied. The LHC experiments can alter the shape and the orbit of each bunch using the quadrupole magnets, thus modifying the instantaneous luminosity to suit their needs. This technique, called luminosity levelling [90, 91], adjusts the beams at the LHCb interaction point to be slightly displaced and not to collide head-on. This is why recorded luminosity at LHCb is currently almost 30 times smaller than at ATLAS [92, 93]. Integrated luminosity is the integral of this measure over time, and thus the total number of events observed is directly proportional to it. The amount of data recorded is often presented in terms of the time-integrated luminosity, which for the LHCb experiment is shown in Figure 2.9.

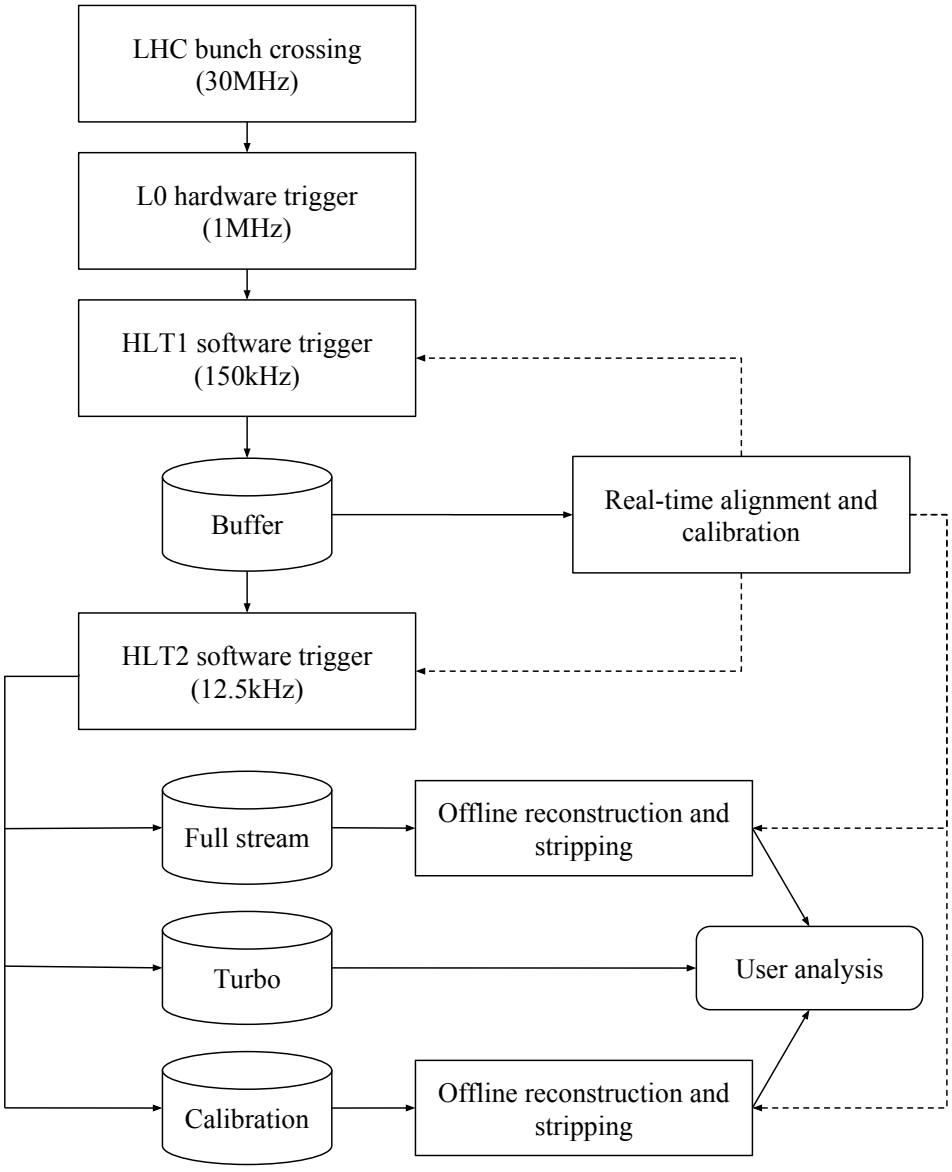


Figure 2.8: The trigger system schema in Run 2 [86].

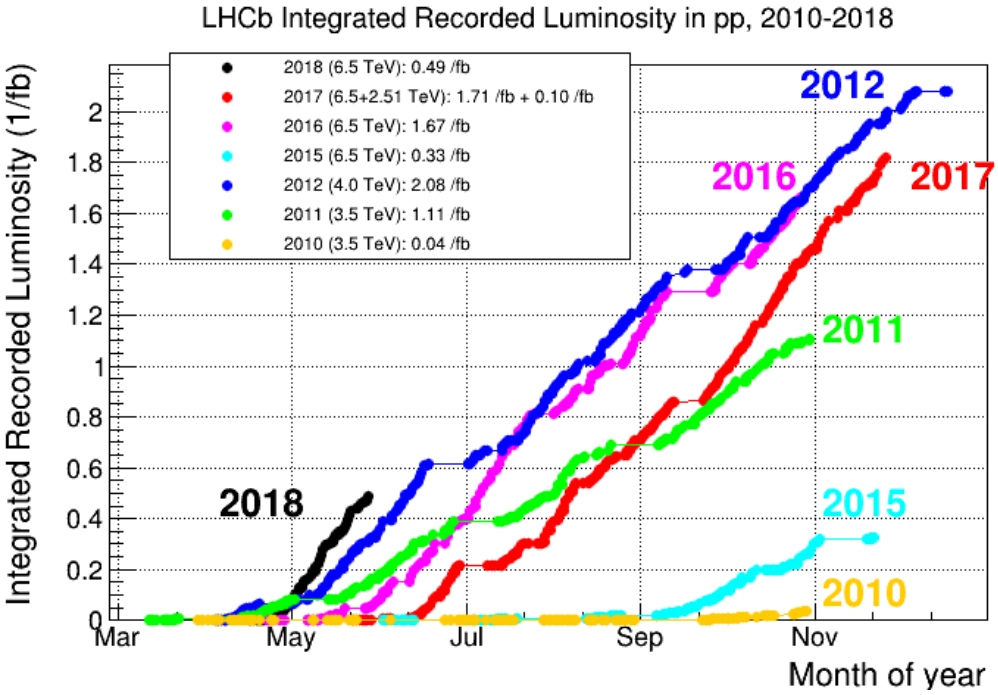


Figure 2.9: Integrated recorded luminosity by the LHCb experiment from 2010 to June 2018 [92].

2.5 The Worldwide LHC computing Grid

The Worldwide LHC Computing Grid (WLCG) [94] is a grid-based computing infrastructure that provides the environment for data processing for the LHC experiments. Both the data production and analysis jobs are executed on the Grid. In addition, the Grid stores and distributes the datasets produced at the experiments.

The Grid is an international collaborative project that consists of over 170 computing centres, which makes it one of the world's largest non-commercial computing grids. It contains four layers, or "tiers", each of which provides a specific set of services. The first copy of LHC raw data is kept on *Tier 0*. The *Tier 0* consists of the CERN Data Centre located in Meyrin, Switzerland and the Wigner Research Centre for Physics in Budapest, Hungary. The two sites are connected by three dedicated 100 Gbit/s data links. There are currently thirteen large computing centres comprising the *Tier 1*, which provides large storage capacity and processing support for the Grid. *Tier 0* distributes both raw and reconstructed data to *Tier 1*, which together then perform data processing. *Tier 2* and *Tier 3* are computing resources typically provided by universities and other scientific institutions for data storage or specific analysis tasks. These sites are mostly located in Europe, North America and Asia, but also elsewhere around the world. Even though the sites are geographically widely separated, they interact with each other

to achieve a common goal to store and analyse the LHC data.

The Grid resources allocated to LHCb are shown in Table 2.2. The storage for the data (on disk and tape) is measured in terabytes, while the CPU is measured in standard HEP-SPEC benchmarking [95]. We can observe from the table a significant increase in used resources. For example, 4,200 CPUs were used in 2009 while today this number is 88,000, representing a more than twenty fold increase.

Tier	**Resource type**	**2018**	**2017**	**2016**	**2015**	**2014**
Tier 0	CPU (HEP-SPEC06)	88,000	67,000	51,000	36,000	34,000
Tier 0	Disk (TB)	11,400	10,900	7,600	5,500	4,000
Tier 0	Tape (TB)	33,600	25,200	20,600	11,200	8,500
Tier 1	CPU (HEP-SPEC06)	250,074	199,113	165,252	139,131	120,948
Tier 1	Disk (TB)	26,252	20,853	15,902	14,041	12,178
Tier 1	Tape (TB)	56,861	41,990	35,044	28,094	11,607
Tier 2	CPU (HEP-SPEC06)	164,109	147,076	88,626	61,181	56,996
Tier 2	Disk (TB)	3,714	3,293	2,719	1,964	1,260
Tier	**Resource type**	**2013**	**2012**	**2011**	**2010**	**2009**
Tier 0	CPU (HEP-SPEC06)	34,000	34,000	21,000	23,000	4,200
Tier 0	Disk (TB)	4,000	3,500	1,500	1,290	991
Tier 0	Tape (TB)	6,500	6,400	2,500	1,800	2,270
Tier 1	CPU (HEP-SPEC06)	92,118	88,891	69,418	42,530	20,216
Tier 1	Disk (TB)	6,997	7,162	3,712	3,254	2,709
Tier 1	Tape (TB)	9,461	5,324	3,878	3,036	3,264
Tier 2	CPU (HEP-SPEC06)	51,772	47,335	40,629	48,308	37,772
Tier 2	Disk (TB)	119	296	211	436	371

Table 2.2: A number of each resource type delegated to LHCb from 2009 to 2018.

2.6 The LHCb Dirac

The LHCb distributed computing on the Grid is performed with the DIRAC software, which allows management and monitoring of the distributed resources. DIRAC allows running thousands of concurrent processes ("jobs") while storing and analysing petabytes of data on the Grid. Even though it was initially created as an internal LHCb project, it was in 2009 released as open source on GitHub [30]. Since then, DIRAC has been adopted by a number of scientific communities, thereby becoming publicly documented and developed.

The LHCb experiment works with an extension of DIRAC called LHCBDIRAC [96], while still using the functionalities that DIRAC provides. LHCb has created two additional systems called the LHCb bookkeeping and the Data Transformation system (for data production

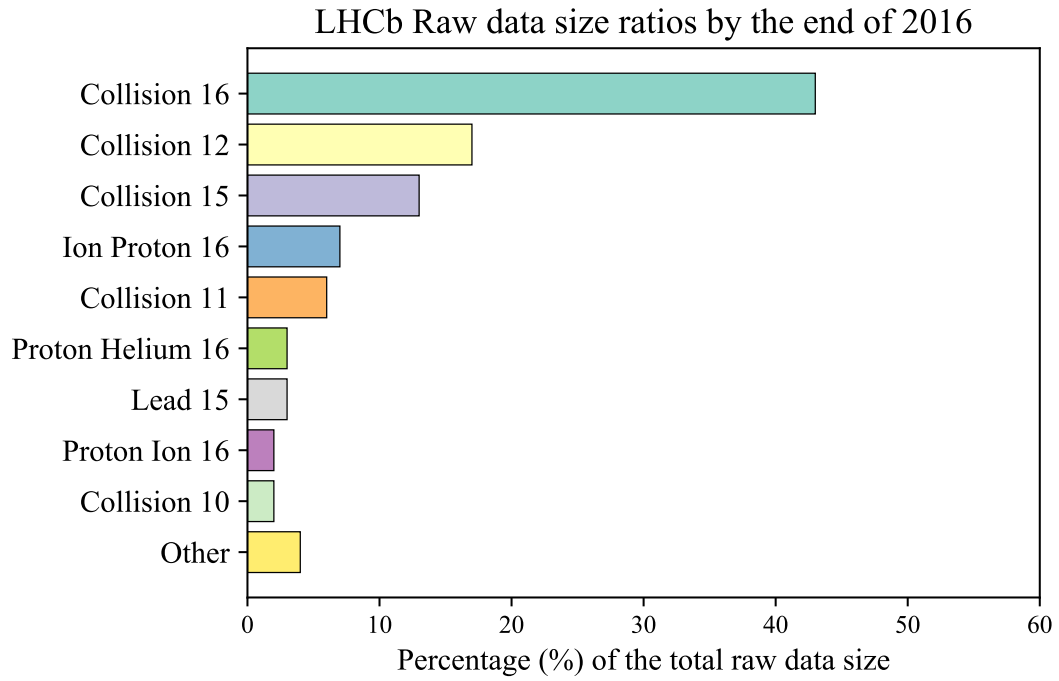


Figure 2.10: Raw experimental data production

management). The LHCb bookkeeping is a metadata and provenance catalogue that documents available datasets and stores information about their location. The Data Transformation system allows for a fully virtualised and autonomous data processing, meaning that it creates and monitors highly sophisticated computational workflows by linking individual steps that process data.

2.7 Data volume and classification

LHCb has accumulated approximately 22.4 PB of raw data from the first data-taking until the end of 2016.[2] Figure 2.10 shows the classification of the data. The label *Collision* stands for the real data with last two digits indicating the year of data collection, for example, *Collision 10* data was collected in 2010. The data that was collected in 2016, *Collision 16*, has a volume of 9.6 PB (43%), which is significantly larger than any of the previous years. The volume of *Collision 15* data is smaller than *Collision 12* because the Run 2 started only in June of 2015.

The label *Proton Helium* in the figure means that the data-taking was conducted with

[2]This information is taken from the DIRAC Monitoring portal with the official web page: `https://lhcb-portal-dirac.cern.ch`

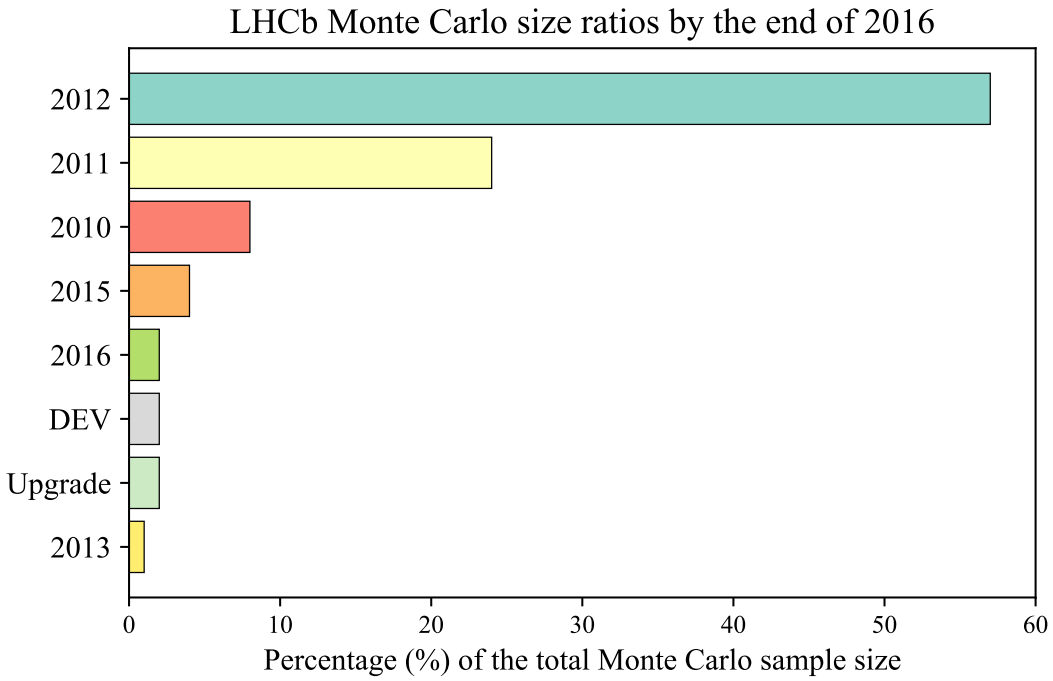


Figure 2.11: Monte Carlo production by the end of 2016. At the time most of the physics analyses were done on Run 1 data. Hence, we can observe that most simulated data corresponds to the 2012 and 2011 detector conditions.

molecules of helium that were hit by beams of protons. These type of experiments are called "fixed" target collisions, and they involve injections of a small amount of gas inside the beam pipe. There have been several *p-gas* (proton-gas) and *Lead-gas* data-taking periods in Run 1 and Run 2. LHCb is the only experiment at the LHC that can operate in fixed target mode.

Monte Carlo simulations mirror the experimental data, and they are used for physics analyses and to understand the detector performance. The LHCb experiment accumulated approximately 10.2 PB of Monte Carlo simulation until December 2016. The classification of the simulation sample is presented in Figure 2.11. Due to the numerous studies being conducted on Run 1 data in 2016, the predominant Monte Carlo sample was from the year 2012 (57%), followed by the 2011 sample (24%). The difference in size of the simulation sample and collision (real) data is significant. However, simulation jobs are much more CPU intensive due to the need to simulate the complex detector geometry and numerous simulation steps. This is why new approaches for "fast simulation" are now developed, which bypass the need for detector simulation [30]. The Monte Carlo workflows are in detail explained in Chapter 4.

The difference in volume of the LHCb experimental data of Run 1 and Run 2 is considerable. During Run 1, approximately 6.5 PB of raw data was collected, and during Run 2 by

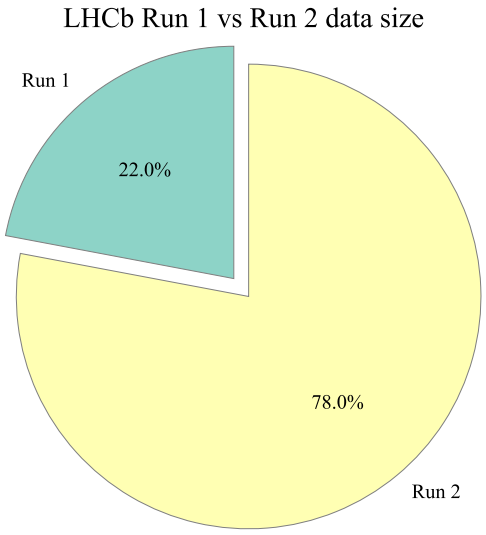


Figure 2.12: The difference in volume in raw data collected in Run 1 and Run 2 by March 2018.

March 2018 almost five times as much (23 PB). This ratio is presented in Figure 2.12.

2.8 The LHCb upgrade

The upgrade plan for the LHC in Run 3 is to increase luminosity and the *pp* centre-of-mass energy to 14 TeV, which will result in generating essentially double the amount of beauty and charm decays inside the LHCb experiment [97]. Thus, it is important for LHCb to increase its data collection rate. However, due to the technical limitations of the hardware trigger, LHCb is unable to accommodate this change without upgrading the detectors. The upgrade of the experiment and the LHC is going to take place during the long shutdown in between Run 2 and Run 3 from 2018 to 2020.

The strategy for the experiment upgrade consists of several significant changes. Firstly, the hardware trigger (Level 0) is going to be entirely removed [23] and replaced with a software trigger. This is due to the fact that the most substantial inefficiencies in the entire trigger workflow currently occur at the first-level trigger [98]. It constrains the readout of the front electronics to 1 MHz, while it is desirable to reach a readout rate equal to the bunch crossing rate of 30 MHz. The redesigned trigger will run on a new event filter farm and is going to be responsible for reducing the full collision rate to accepted output rate to be sent to storage.

The second significant change is the upgrade of the LHCb sub-detectors. For instance, the silicon sensors will be redesigned in the tracking sub-detectors. Their location (VELO, TT

stations, IT and OT) will stay the same, although the sub-detectors themselves will be replaced. The four TT planes will be replaced by new high granularity silicon micro-strip planes with improved coverage of the LHCb acceptance. The three TT tracking stations will be replaced with a scintillating fibre tracker [99]. Finally, the VELO must maintain or improve its performance while delivering readout at 30 MHz in the operating conditions of the upgrade. It will be entirely replaced by a detector based on hybrid pixel sensors [100], but it will reuse large parts of the current mechanical infrastructure, such as the cooling system.

The data volume in Run 3 will drastically increase. During Run 2, a new processing strategy called Turbo (introduced Section 2.4) that is currently used for about 30% of the data stream, is going to be the primary data processing method in Run 3. This means that the data is going to be reconstructed and preselected in "real-time" through the turbo stream, and there is not going to be any (or very little) offline raw data processing.[3] Due to the smaller size of the recorded events, LHCb will be able to store a much higher number of the events via turbo stream comparing to the full stream.

To discuss the expected data volume in Run 3, we use a unit that describes the length of time of the LHC run. In one year of Run 3, we estimate that there is going to be from 1400 hours to 1950 hours of the LHC running.[4] The bandwidth of the data processed by the trigger is $2\,\mathrm{GB\,s^{-1}}$ to $5\,\mathrm{GB\,s^{-1}}$.Therefore we can estimate the amount of data collected in one year of Run 3 to be up to 35 PB, leading to approximately 100 PB in three years of running. The drastic increase in data volume will inevitably result in higher complexity in physics analyses, meaning that writing efficient and scalable software for new processes will become more challenging.

2.9 Chapter summary

In this chapter, I described the experimental setup of LHCb. I discussed the process of data-taking and the volume of data recorded by the experiment. I mentioned future aspirations and plans for the experiment's upgrade. In the next chapter, the essence of the LHCb preservation strategy is introduced.

[3] Monte Carlo simulations and production will be done offline.

[4] This number was estimated using the "stable beam time" from the LHC performance information in 2016 [101] and 2017 [102].

Chapter 3

The LHCb preservation strategy

The LHCb preservation strategy is a necessary schema to ensure that the information required to perform data analysis is adequately recorded for the long-term future. In this chapter, I introduce the components for the correct interpretation of the LHCb data and a strategy for their preservation. The components include the experimental data itself, the LHCb software, conditions data and documentation.

3.1 Strategy for the data

The preservation of the LHCb data itself covers the storage and the curation of the datasets. It also ensures the correct interpretation of the data formats for both the raw and derived datasets. These efforts are already underway, i.e. by keeping multiple copies of a dataset at different sites, thus ensuring safe storage. The raw files are stored in multiple copies: always at least one copy at CERN and the others at major LHC Grid sites (*Tier 1*). Regarding the derived datasets (ones produced after the reconstruction and stripping), typically two (or three) copies of each file are saved on the Grid, and one copy is sent to the CERN Advanced STORage management (CASTOR) [103] to be saved on tape. CASTOR is a hierarchical storage management system managed by the CERN IT group. Files can be stored, retrieved and remotely accessed using command-line tools or user applications. The main disk-based storage system used at CERN is called EOS. In addition to storing physics data and simulation, it also provides storage space to individual users to perform data analyses. The service currently uses disk space located at the two CERN *Tier 0* centres, in Meyrin and Budapest [104].

The physical preservation of the recorded files is a significant endeavour by itself. All

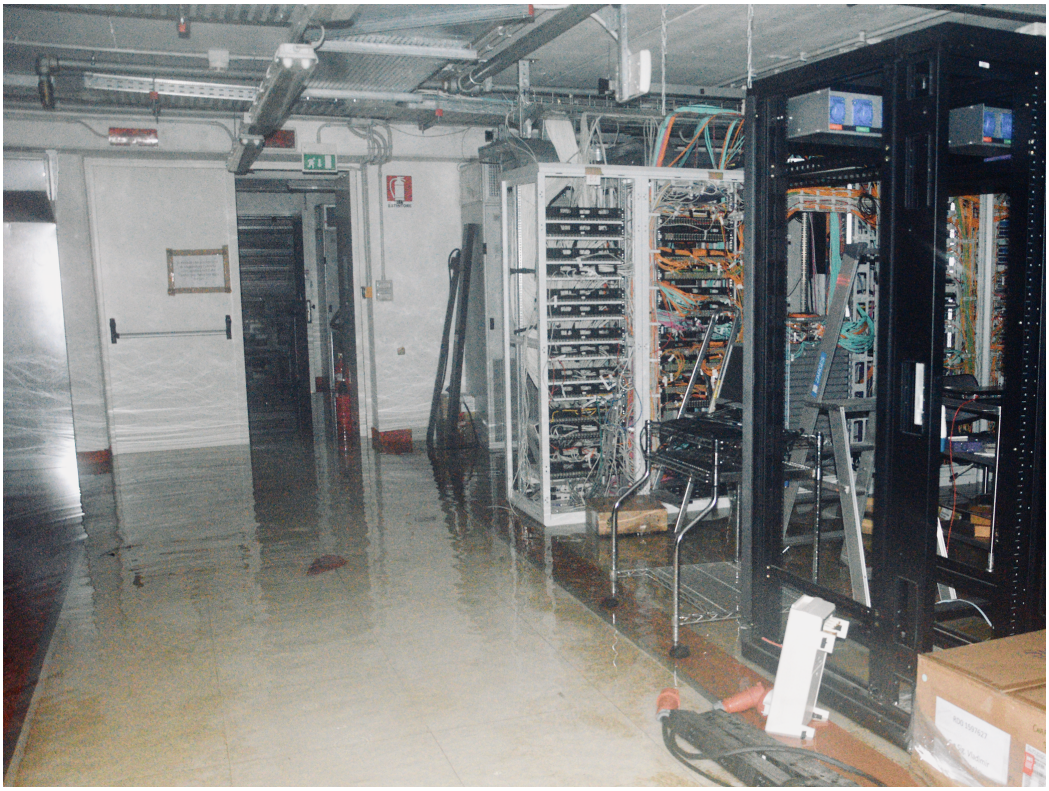


Figure 3.1: One of the flooded rooms in CNAF. Photo credit INFN-CNAF.

storage media degrade with time, hence the storage needs to be monitored and tested at regular time intervals. Furthermore, hardware manufacturers may become obsolete, leaving the existing hardware without technical support and updates. On the scale of decades, hardware migrations need to be planned. This effectively means that before one type of storage becomes unresponsive or obsolete, data needs to be copied to the new media. The first hardware migration at LHCb was organised in the first long shutdown of the LHC, and during this time the data was copied from old tapes to new ones [105].

Disasters that may damage or destroy the data are rare, but they do happen. For example, on 9th November 2017, a water pipe broke in front of the *Tier 1* Computer centre site (CNAF) in Bologna, Italy.[1] The computing centre that was located underground was flooded, and all services that it provided were affected including the storage systems (disks and tapes). A photograph taken on the site is shown in Figure 3.1. Recovery from the disaster lasted for the next four months, and the site was brought back in March 2018. Even though around 4 PB of the LHCb data was destroyed, the data loss was reported to be "minimal" as no raw events have been lost and the damaged datasets could have been reproduced or copied from other locations.

Due to the large datasets collected at LHCb, storage efficiency is highly desired. Effi-

[1] The centre was provided by the national centre of Italian Institute for Nuclear Physics.

ciency regarding writing and reading time is essential as the LHCb trigger transmits a high rate of events during data collection. Therefore, LHCb data file format needs to be concise and fast to transfer to conform to the performance and storage constraints. Efforts to decrease file sizes can lead to significantly reducing the need for storage space.

There are three groups of file formats used at LHCb to capture raw, derived and analysis data respectively. Their relationship in terms of technical capability and semantic complexity is shown in Figure 3.3. The raw data formats are used to capture information about the detector readout. After data processing, the event file format becomes more complex and the information that they keep, such as kinematic information of particles, more sophisticated.

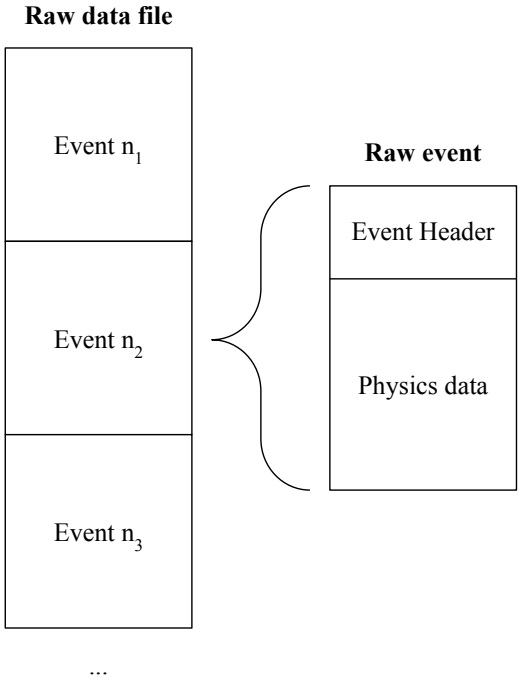


Figure 3.2: The structure of raw files. The file consists of a stream of events, where each event is defined by a header and a data block describing physics data. The header contains information like the checksum of the event.

3.1.1 The raw file format

The raw data is kept in a format called MDF (stands for Mast Data Files, also referred to as “RAW”), which stores the raw events from the HLT. A raw event is the concatenated set of data banks containing information that comes out of the detector subsystems [106]. The banks are combined in the online data acquisition system, shipped through the trigger and eventually written to an MDF file.

The MDF file format is streamable (or "splittable") meaning that events are simply stored one after another (as shown in Figure 3.2). This file format was implemented due to the data-taking requirement to individually transfer the physics events from the experimental site. The streaming nature of the file format also provides a streamlined way to merge the files as they can be simply concatenated together. This is one of the main differences in data formats between raw data and processed data.

3.1.2 The ROOT file format

The ROOT framework [107] is a free and open source data analysis software developed at CERN. It provides various functionalities for data processing, statistical analysis, visualisation and storage. It is mainly implemented in the C++ programming language, but it is also integrated with other languages such are python and R.

The ROOT file format is commonly used for data production and for individual data analyses at LHCb and in the experimental HEP community in general. Both the ROOT application and the file format are being continuously maintained by a team of developers based at CERN. Even though its use has many advantages, such as a good compression, fast input/output data transfer [107] and a large spectrum of plotting options, from the preservation point of view it has several disadvantages. For example, its custom encoding can only be read with the ROOT application. In addition, it is used by a relatively small academic community and it is relatively unknown outside of high-energy physics. This may be disadvantageous considering that projects with large user communities and industry support are more likely to be adequately maintained in the long-term future.

3.1.3 The DST file formats

The management of both real and simulated events in the LHCb software is achieved through the use of the LHCb Event Model [108]. It is a set of C++ classes that capture information such as particle hit propagation through the LHCb tracking stations, particle energy, momentum, the entry and exit points, etc. For MC production, they also include Monte Carlo truth information, meaning that an event was created in a simulation. The GAUDI Transient Event Store (TES) is used to exchange event information during data processing. Applications retrieve their input data from the TES and publish their output data in different locations. This is possible as the event data is logically subdivided into a tree structure analogous to a Unix file system. In the end, the LHCb Event Model is saved in the "Data Summary Tape" (DST) file

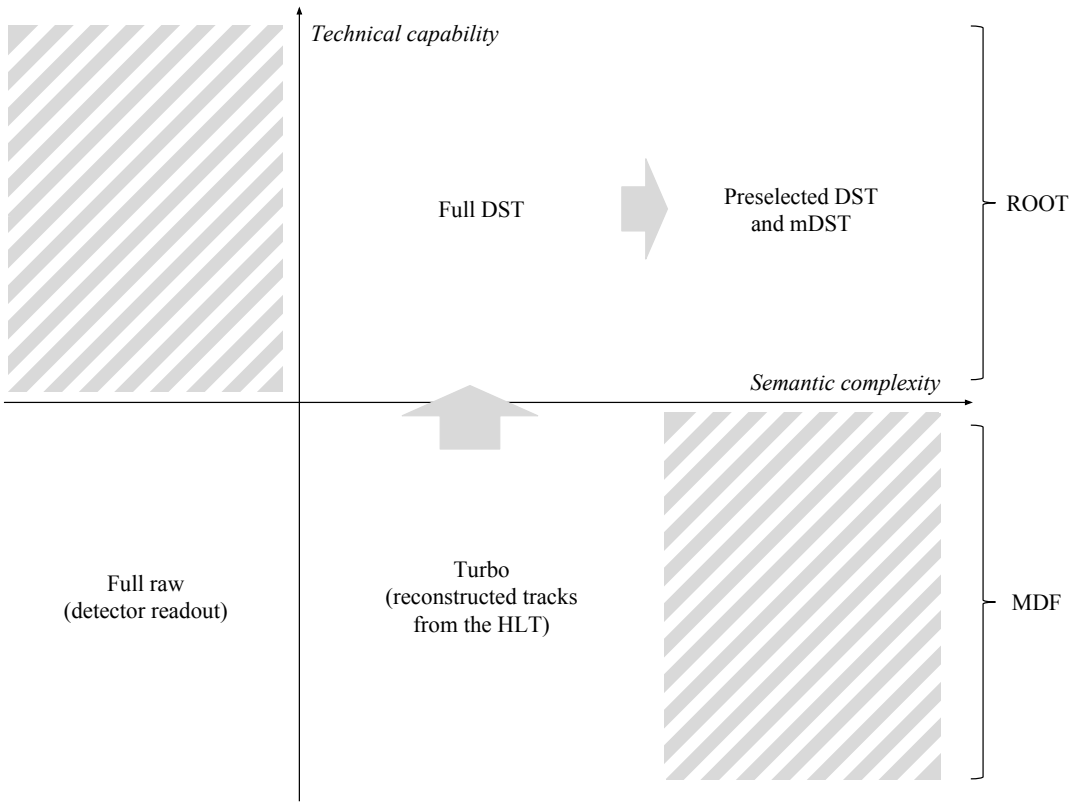


Figure 3.3: The LHCb data formats presented according to their semantic complexity and physics capacity. This figure relates to real data formats, as Monte Carlo sample is always stored in ROOT-based file formats.

format.

The DST file format is a ROOT-based file format used to store data after the reconstruction and preselection of either MC simulation or real data. To decrease the disk space used to store many copies of the output data, a new file format, MicroDST (mDST or uDST), was created. It only stores the selected candidates, thus it is often an order of magnitude smaller than a DST [109]. Any DST-based file format can be interpreted by the ROOT framework using a set of LHCb dictionaries. This compatibility is shown in the first row (labelled ROOT) in Figure 3.3. Full DST captures information that was created after the reconstruction, which is reconstructed particle tracks, particle identification (PID) values, primary vertex location etc. In preselection, the DST or mDST files include signal particles that were selected by the algorithms, and they, therefore, have a higher semantic complexity.

3.2 Strategy for the conditions databases

Physical preservation of the data is however not enough, as its correct interpretation in the long-term future also needs to be ensured. A number of databases are needed to reconstruct and analyse the LHCb physics events. These databases provide a complete description of the detector state and running conditions at the time of data collection. The conditions databases are the following:

- DETECTOR DESCRIPTION DATABASE (DDDB) captures the detector geometry.
- ONLINE database captures time dependent conditions of the experiment, like alignment and calibration constants of the LHCb subsystems.
- LHCBCOND is a manually updated database that contains information about the experimental subsystems that does not change in time during data collection.
- DQFLAG is used to flag a dataset that does not agree with expectations as "bad", indicating that one of the subsystems had not worked properly.
- SIMCOND is a MC equivalent to both the ONLINE database and LHCBCOND to be used for MC simulations.

The conditions data is stored in the XML [110] format inside the GitDB database.[2] The data volume of these databases is low (in the order of gigabytes), so there is no major concern about the storage.

3.3 Strategy for the software

The software developed by the LHCb collaboration to reconstruct, select and analyse data can be broadly classified into two categories: the production software and the analysis software. The production software is developed under a central LHCb effort. It includes all the standard tools needed to record and reconstruct the physics events. It has to follow strict procedures, and it is validated by the LHCb computing group before each release. On the other hand, the software used in physics analysis is developed by the analysts. Once they retrieve data using the production software, they are free to use any tool and method to analyse the data. These tools and methods can differ between various working groups and analyses.

[2]The five databases are stored in the following GIT repository: `https://gitlab.cern.ch/lhcb-conddb`

3.3.1 The production software

The LHCb software is the most critical component in the LHCb data processing. It is used in a range of different processing environments, from real-time event processing in the trigger and the data and Monte Carlo production, to advanced physics analyses.

The LHCb software is based on GAUDI [111], the object-oriented framework designed to provide common infrastructure and environment for the software applications of the LHCb experiment [112]. It was created prior to LHC data-taking back in 1998. Its development was driven by LHCb, but it was also adopted by the ATLAS experiment [113]. Gaudi was successfully used throughout Run 1 and Run 2, and it allowed development of many applications for data processing and physics analysis.

The LHCb software contains several millions of lines of C++ and python code. Projects of this scale are managed using recognised software engineering practices including a version control system for the source code, continuous integration and code reviews. The LHCb software projects can be sorted into the following categories according to their role in data collection and processing.

- **Applications for distributed processing:** DIRAC, LHCBDIRAC, LHCBVMDIRAC, VMDIRAC, LHCBGRID, BEAUTYDIRAC
- **Applications for detector monitoring:** VETRA, ORWELL, LOVELL, PANOPTES
- **Applications for data processing and high-level physics analyses:** MOOREONLINE, MOORE, BRUNEL, DAVINCI, URANIA, PANORAMIX, ALIGNMENTONLINE, ALIGNMENT, BOOLE, GAUSS, BENDER, KEPLER, VANDERMEER, ERASMUS, CURIE, NOETHER, CASTELAO,
- **Framework:** GAUDI, ONLINE, GEANT4, LHCB,
- **Component or library:** HLT, STRIPPING, ANALYSIS, LBCOM, PHYS, REC

Most notable applications are: the reconstruction application BRUNEL [114], the trigger application MOORE [115], the preselection application DAVINCI [116], the applications for Monte Carlo simulations GAUSS [117] and BOOLE [118], and the event and detector visualisation program PANORAMIX [119]. The other applications are also used for data production and analysis, but they will not be discussed in this thesis.

The list of the projects, together with their size and the number of versions is shown in Table 3.1. Projects are independently developed and released within LHCb, and so far more than a thousand project versions have been published. Every project version is tagged with a *major* and *minor* release number. For example DAVINCI V33R1 or GAUDI V23R5. If published projects have issues that need to be fixed, the solutions are released in the form of a "patch". A patch is a piece of software designed to update or improve the project. It is denoted with a suffix *pNo*, for example DAVINCI V36R7P3. A patch does not introduce any major changes to the project.

Experiment-specific information, such as the LHCb Event Model and the Detector Description, are provided within the GAUDI framework as core software components. The framework together with these components and applications constitutes the LHCb software stack. The software stack is modular, which means that each project is dependent on a number of other projects. Essentially, in order to compile a high-level application, such as for example the physics analysis application DAVINCI, the projects that are below in the software stack need to be compiled.

Regarding the technical implementation, the preservation of the software requires:

1. The version control system with all versions of the code. This is necessary to help understand the code evolution, issues, etc.

2. The documentation associated with the code. This means documentation in the code itself, guides, manuals, tutorials and web pages.

3. The binaries used for production, as well as the possibility to reproduce and rerun them.

Regarding the preservation of the version control system, the migration to a future solution needs to be planned, as it is crucial for the preservation in the long term. An example of a transition scheme at LHCb is the migration from Concurrent Versions System (CVS) to Subversion (SVN), and then the second migration to the GIT version control system.

A solution for the preservation of the binaries may already exist, as after compilation, the LHCb software is deployed to the CERN Virtual Machine File system (CVMFS) [120]. CVMFS is a read-only, free and open source file system designed for efficient software distribution [121, 122]. It can be mounted on any Linux- or Unix-based platform and it allows easy software deployment to the LHC Grid sites. Every released version of the LHCb software stack with its components is installed on CVMFS.

While the pure preservation of the binaries is not a difficult task when it comes to the several terabytes installed on the CVMFS system, making sure that the necessary runtime environment is available is a much harder one. In recent years, many improvements in the virtualisation technology made this possible, which is explored in Chapter 5. However, for the LHCb data processing, information about the compatibilities between data and software needs to be preserved. As currently, this information is not readily available, a provenance database, introduced in Chapter 4, was designed and implemented to capture this metadata. The database gathers information about the LHCb data and associated software to provide a global view of what is required for data preservation.

3.3.2 The analysis software

The analysis software used by the LHCb physicists is in essence very different from the production software. To encourage freedom to innovate, the physicists use custom methods, which resulted in the development of a large number of analysis tools (and different versions of the same tool). The analysis software is strongly tied to particular derived datasets and analysis goals, making it dependent on many factors. Thus, its preservation needs to be implemented in a flexible way that is sympathetic to the existing practices and is a natural part of the research workflows. This is discussed in detail in Chapter 8.

3.4 Strategy for the documentation

Clear and complete documentation is essential for understanding the LHCb data, software and analyses in the future. Basic information of physics analyses is recorded in the LHCb working group database [123], which provides links to further documentation at the LHCb TWIKI [124] portal and the CERN Document Server (CDS) [125]. The TWIKI is an internal web-portal that represents the main tool for analysis documentation. The TWIKI pages contain information such as analysis description, presentations, data provenance and sometimes a link to the analysis software. CDS preserves pdf documents of physics analyses such as technical notes, papers, theses etc. The TWIKI and CDS also contain documentation about the LHCb data, experimental software and the detector. Finally, there are collaborative tools like LHCb Q&A [126] that provide questions and answers about various uses and problems of the LHCb software, and thus, supplement the primary documentation. The preservation and transition scheme of these services is planned by CERN and LHCb [30].

3.5 Chapter summary

In this chapter, I presented the high-level preservation strategy of LHCb. In the following chapter, I introduce the LHCb data production workflows. I present the design and implementation of a graph database that captures data and software dependencies, that were previously obscure. Linking these resources together facilitates their preservation and future reuse.

Software projects	Size	Number of versions
ALIGNMENT	9.5 GB	24
ALIGNMENTONLINE	1.1 GB	14
ANALYSIS	103 GB	78
BEAUTYDIRAC	9.4 MB	5
BENDER	452 MB	28
BOOLE	11 GB	32
BRUNEL	4.3 GB	66
CASTELAO	498 MB	2
CURIE	5.8 MB	2
DAVINCI	113 GB	143
DIRAC	1.4 GB	49
ERASMUS	3.2 GB	28
GAUDI	74 GB	65
GAUSS	61 GB	67
GEANT4	35 GB	43
HLT	58 GB	85
KEPLER	761 MB	5
LBCOM	64 GB	92
LHCB	260 GB	98
LHCBDIRAC	2.4 GB	63
LHCBGRID	30 MB	19
LHCBVMDIRAC	33 MB	20
LOVELL	1.1 GB	5
MOORE	14 GB	132
MOOREONLINE	2.5 GB	37
NOETHER	79 MB	5
ONLINE	73 GB	86
ORWELL	1.1 GB	11
PANOPTES	3.3 GB	17
PANORAMIX	3.6 GB	13
PHYS	149 GB	103
REC	267 GB	95
STRIPPING	116 GB	103
URANIA	4.9 GB	13
VANDERMEER	229 MB	7
VETRA	4.5 GB	11
VMDIRAC	7.1 MB	2

Table 3.1: A list of LHCb software projects with their total sizes and number of versions. Counted on 29 May 2018. using commands from Ref. [1].

Chapter 4

Provenance database

Data provenance is essential for the preservation of datasets because it indicates how a dataset was created and aids in recovering it. In this chapter, I explain how we assist the data preservation effort by making the LHCb data provenance more understandable and accessible.

Firstly, I introduce the LHCb data production and the current provenance management system in detail. Then I explain how our solution, the provenance database, completes the picture of the data preservation efforts [2]. I provide details on the design and implementation of the database to capture data provenance, emphasising its link to the physics analyses. The provenance database gathers information about the real data and simulation productions, linking it to the dependencies in the LHCb software stack. Finally, I present a variety of studies on the database to draw valuable conclusions for the preservation efforts.

4.1 Capturing data provenance

The term provenance implies ownership, custody or location of a historical object [127]. It was originally used to describe a piece of art, but today it is used in a wide range of fields including natural sciences and computing. Computational provenance captures where data came from, how it was derived, manipulated, combined and changed over time [128].

In order to properly utilise the LHC data, it needs to be associated with metadata describing its origin. According to the FAIR (Findable Accessible Interoperable and Reusable) [129] principles, data provenance provides a formal model for this purpose. It can be presented as a directed, acyclic graph where interactions are recorded as a set of *edges* that relate data-items. This model has been standardised by the World Wide Web Consortium (W3C) as the PROV

data model [130]. Although provenance can be understood in simple terms through its visual representation (shown in Figure 4.1), in practice, the provenance of data in complex HEP experiments contains thousands of nodes and edges.

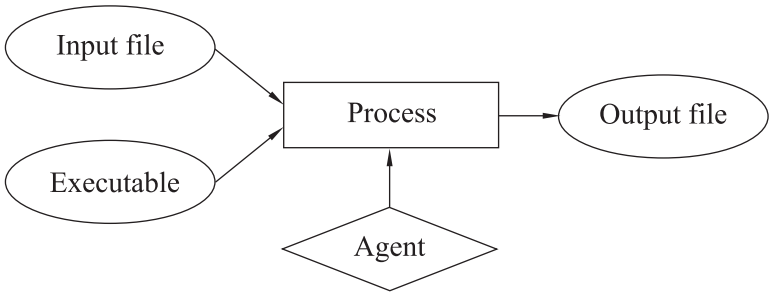


Figure 4.1: A simple visual representation of a data provenance graph.

In order to organise the LHCb metadata in a provenance graph that will facilitate data preservation, the metadata needs to be collected from various sources at different locations within the LHCb infrastructure. Much of the data provenance is defined during data collection (such as beam and detector conditions), but also during the data processing phase, when the collision events are reconstructed to reveal information about the particles.

Before I proceed with the design and implementation of the database, it is necessary to define each stage in the LHCb data production, which is a process of reconstructing and preselecting the collision data before it is released to the analysts. Data production consists of several stages of transformation, and each stage is described by a set of instructions called a *Step*. The term data production can refer to both the processing of the experimental data or the creation of simulation samples.

4.2 Processing of the experimental data

The raw experimental data contains the full event readout from the LHCb detector. Before it can be analysed, the data goes through a chain of transformations managed by the LHCb computing group, during which, the detector readout is used to identify particles that traversed the detectors and reconstruct their trajectories.

There are two approaches for the experimental data production in the second run of the LHC (as shown in Figure 4.2). The first approach consists of two stages: the reconstruction and the preselection. The second approach processes the data directly from the trigger system through a turbo stream, which was introduced in Chapter 2.4.

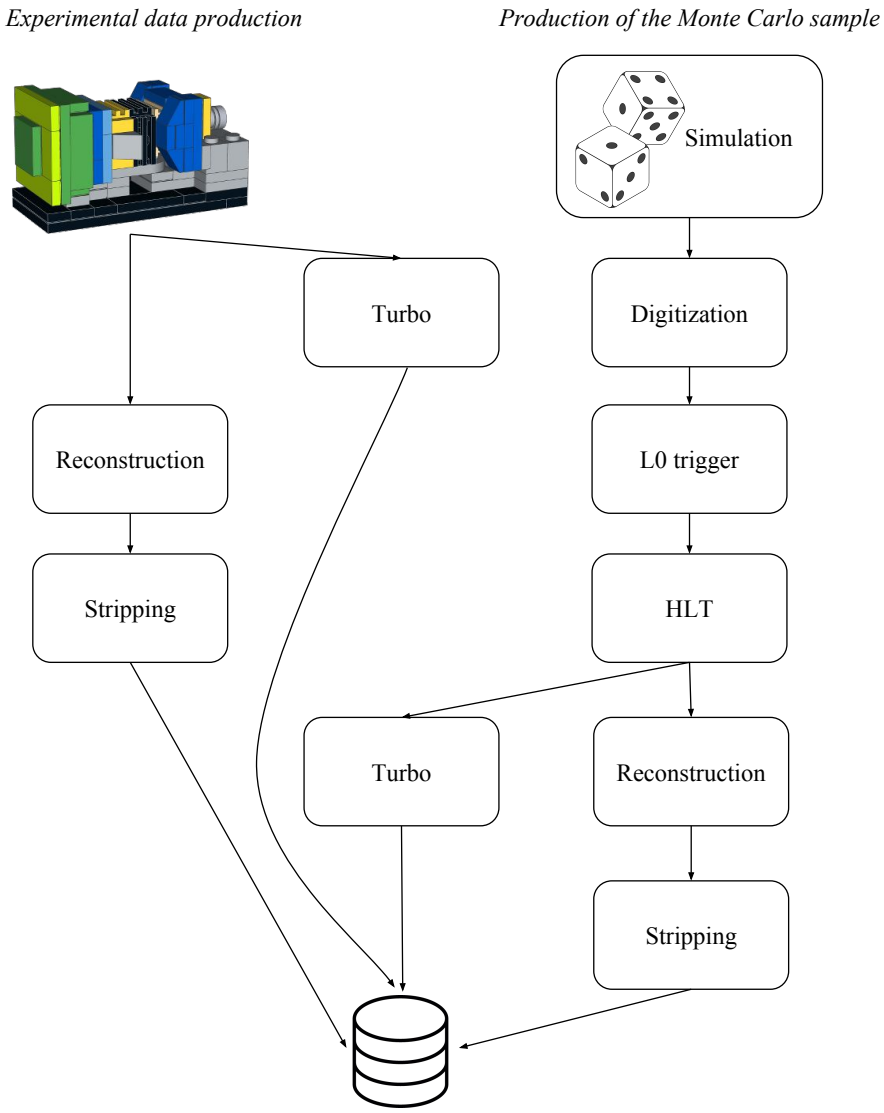


Figure 4.2: Schema of the Monte Carlo and experimental production workflow.

4.2.1 Reconstruction

The reconstruction step processes raw experimental data that was selected by the trigger system using the BRUNEL application [131]. It reconstructs tracks of the charged and neutral particles, computes PID values and locates the primary vertex position.[1] The output of BRUNEL are fully reconstructed events stored in the full DST file format (introduced in Section 3.1).

4.2.2 Preselection

The preselection (also called stripping or streaming) takes place after the reconstruction stage.[2] The DAVINCI application [131], employed in this stage, uses objects produced by BRUNEL to reconstruct particle decays, create composite particles and locate secondary vertices. The program then calculates a wide variety of kinematic quantities of decays, such as invariant masses of mother particles, distances of flight, decay times etc., to be used as part of the preselection to further filter the data.

The preselection takes place because the size of datasets after reconstruction is too large for individual use. To specify what selection will be performed in this stage, the LHCb computing group receives requests from the analysis working groups. These requests contain *stripping lines*, which are loose sets of filters about the final state particles that select particular decays or physics processes of interest. All stripping lines are then run centrally over the reconstructed datasets, and only events that pass the selection criteria of a stripping line are available to be used in analyses. The preselection reduces the data volume to a manageable size for data analyses, and its output is kept in DST and mDST files. Typically, the output of a stripping line must contain less than 0.05% of the total number of events if the full event information is saved [89].

The preselection is done for the first time as the data is collected. It is afterwards repeated to take advantage of new developments in the selection algorithms, which may fix previous problems, introduce new stripping lines or improve existing lines. A new stripping version typically replaces the old one, as it then becomes the "recommended" one for use in physics analyses. However, there is *incremental stripping* that adds to the previous selection, and in this case both original and incremental stripping are kept as recommended versions.

[1] The primary vertex is the *pp* interaction point and its accurate estimation is essential for precise measurements at LHCb [59].

[2] A reconstruction or preselection step in data production is usually referred to as a *processing pass*.

4.2.3 Merging

Data processing jobs, due to the size of datasets, are typically split into subjobs and distributed on the Grid computing resources. Each of the subjobs handles a fraction of input data and creates an output file following the job description, whether that is selection, reconstruction or simulation. By distributing input data to many subjobs, the total time of job execution is much shorter. The subjobs are independent of each other, meaning that if one subjob fails it will not influence the others, and it could be rerun at a later stage.

The output files are produced across the network of computers, and some of them can be fairly small in volume. In order to ease their management and not to clutter the LHCb bookkeeping system, they are collected and merged into datasets of a standard size (approximately 3 GB). The merging stage does not directly change the events, it merely gathers them together, maintaining the data format and structure.

After the preselection and merging, data production for the real data is complete. The datasets are distributed onto the Grid, where they are made accessible to the members of the LHCb collaboration.

4.3 Production of the Monte Carlo samples

The Monte Carlo (MC) simulations play an essential role in many studies at LHCb. Several software applications within the GAUDI framework are used in the generation of the MC samples, to make sure that they mirror the real data as much as possible. The Monte Carlo production workflow [131] consists of several stages, as shown on Figure 4.2.

It is important to note that starting from the trigger selection, processing of the experimental data and Monte Carlo samples is exactly the same. For these processes, the same software applications and configurations are used. However, Monte Carlo events contain, besides the particle hit information, an extra “Monte Carlo truth” information. The truth information records the physics history of the event and the relationships of hits to incident particles. This history is propagated through to subsequent steps in the processing so that it can be used in physics analyses. Simulated raw datasets are thus larger than real data, even though the format of the file is identical to that of real data.

4.3.1 Event generation and simulation

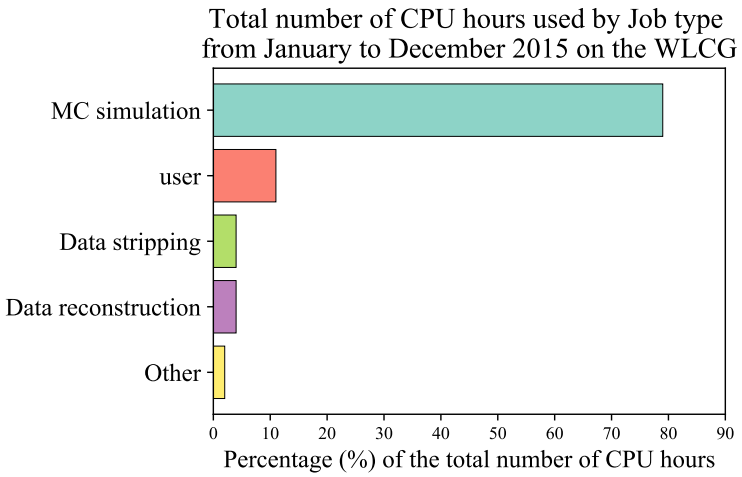
Event generation is the first stage of the MC simulation production. In this stage, the type of particles and their decays are specified, so that events of particular interest can be generated. The GAUSS application [131] simulates the events and the behaviour of the LHCb detector to allow understanding of the experimental conditions and performance. This is done in two independent phases, using various packages. The first phase consists of the generation of *pp* collisions and the particle decays in channels of interest for the LHCb physics program. This process is done using packages like PYTHIA [132] to model *pp* collisions and the particle production, and a specialised package for B-decays such as EVTGEN [133] (among others). This phase also handles the simulation of the running conditions, as both packages have been tuned for the particle production and decay inside the LHCb detector.

The second phase of the GAUSS execution consists of the tracking of the particles in the LHCb detector produced by the generator phase. The simulation of the physics processes, which the particles undergo when travelling through the experimental setup, is delegated to the GEANT4 [134, 135] toolkit. The output of GEANT4 captures the hits produced in the detectors, which is then converted into the LHCb Event Model. The simulated samples are recorded in the SIM data format, which is a ROOT-based data format used to capture the LHCb Event Model that was produced in GAUSS.

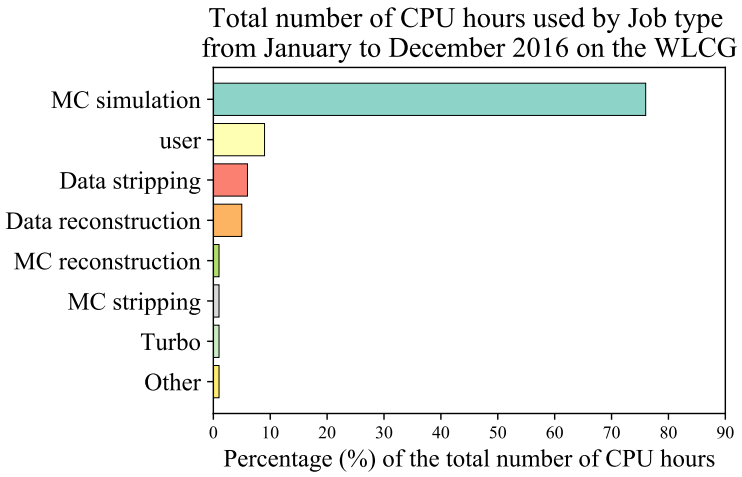
Whilst during data collection the online trigger is the most CPU-intensive process, in the offline data processing the most computationally demanding processes are the Monte Carlo simulations. The total numbers of CPU hours used by job type can be seen in Figures 4.3a, 4.3b and 4.3c (in 2015, 2016 and 2017 respectively). The event generation (labelled as MC simulation stage) takes more than 3/4 of the CPU time on the LHC computing grid, most of which is allocated by GEANT4. From the moment the Monte Carlo production requests are submitted it can take up to six months to receive the samples, due to a large number of production requests and the time it takes to produce them.

4.3.2 Detector response

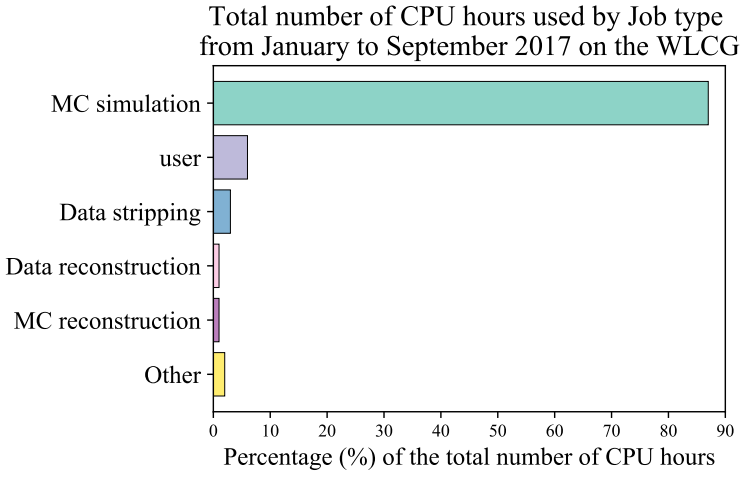
The second stage in the Monte Carlo production is called digitisation. The application BOOLE [131], based on the GAUDI framework, reads in the output of GAUSS and produces the digitised detector readout using the information from GEANT4. It simulates the response of the real detector and produces files in a ROOT-based format called DIGI, which is an MC



(a)



(b)



(c)

Figure 4.3: Total number of CPU hours used by Job type in 2015, 2016 and 2017 on the Worldwide LHC Computing Grid (WLCG)

equivalent to MDF and captures output that mimics raw data banks.

4.3.3 Trigger and turbo emulation

The application that runs the trigger response is MOORE [131]. It filters the simulated data in the same way the real data is filtered at the LHCb experiment. First, the L0 trigger simulation is executed and then the HLT.

The turbo stream is also included as a part of the Monte Carlo production to obtain the simulation samples in the same file format as real data from the turbo stream. The application used to process the turbo stream, TESLA, writes out a compact summary of physics objects containing information necessary for data analyses. The TESLA application is designed to handle both offline and online data processing in an identical way. This allows running the same algorithms offline as are run in the HLT environment.

4.3.4 Reconstruction and preselection

The BRUNEL application processes the simulated sample in the same way as it processes the real data, independently of the Monte Carlo truth information. Therefore, the same algorithms are always executed on both the real data and the simulated sample.

Finally, the Monte Carlo sample is subjected to exactly the same stripping selection as real data. This stage is executed using the same methods and applications as described in Section 4.2.2. It is essential to use the same stripping lines on both the real data and the Monte Carlo sample, to ensure consistency between the final outputs.

4.4 The LHCb bookkeeping

The LHCb bookkeeping database documents metadata of the available datasets and stores information about their location. It is the starting point for every physics analysis, as it allows the analysts to browse through the catalogue to find the datasets that they need.

The catalogue is organised in a tree-like structure, and the branches of the tree present a path to the dataset location, which is referred to as *the bookkeeping path*. Each bookkeeping path contains partial information about the origin of the datasets, such as for example whether it captures real data or simulation.

Example 4.4.1. The path provides this information in the following way. The first example shows the path description in the catalogue, and the second and third show real data and Monte

Carlo paths respectively.

```
# Path description
/<origin>/<year>/<conditions>
      /<processingPass>/<eventType>
          /<fileType>

# Real data
/LHCb/Collision15/Beam2510GeV-VeloClosed-MagDown
      /Real Data/Reco15a/Stripping22b
          /90000000

# Monte Carlo sample
/MC/2012/Beam4000GeV-2012-MagDown-Nu2.5-Pythia8
        /Sim08a/Digi13/Trig0x409f0045
            /Reco14/Stripping20
```

The system stores a catalogue that maps the bookkeeping paths to the locations of the datasets. This is implemented in two levels, where the first one is the logical file name (address) (LFN) and the second one comprises of the physical addresses of the datasets (PFN). The LFN is resolved to reveal the physical location of the datasets on the Grid.

Example 4.4.2. An example of the LFN and PFN mapping is presented in the following. The example shows logical and physical paths to a Monte Carlo sample produced in 2011. In this example, the physical location of the sample is at CERN on the EOS storage system, and it is specified in a format that can be used to access the file through the internet.

```
LFN: /lhcb/MC/2011/ALLSTREAMS.DST/00024917/0000/
00024917_00000024_1.allstreams.dst
PFN: root://eoslhcb.cern.ch//eos/lhcb/grid/prod/lhcb/MC/
2011/ALLSTREAMS.DST/00024917/0000/
00024917_00000024_1.allstreams.dst
```

We note that there is no direct link between the bookkeeping path of a dataset and the production file that describes how it was produced. Therefore, the path itself is not sufficient to

```
Type: Reconstruction
State: Done
Event type: 90000000 Full stream
Number of events: -1
Configuration: LHCb version: Collision16
Conditions: Beam6500GeV-VeloClosed-MagUp
Processing pass: Real Data
Input file type: RAW
DQ flag: OK

Processing Pass: Reco16
Step 1 FULL-Reco16-cond-20161004-RDST(130084/Reco16):
        Brunel-v50r3
System config: x86_64-slc6-gcc49-opt
Options: $APPCONFIGOPTS/Brunel/DataType-2016.py;
        $APPCONFIGOPTS/Brunel/rdst.py
DDDB: dddb-20150724 Condition DB: cond-20161004 DQTag:
Extra: AppConfig.v3r284;SQLDDDB.v7r10 Runtime projects:
Visible: Y Usable: Yes
Input file types: RAW(Y)
Output file types: BRUNELHIST(Y),RDST(Y)
```

Figure 4.4: An example of a production file.

reproduce the LHCb data production of a particular dataset. This is an issue in regards to the data preservation effort, which needs to be addressed in the provenance database.

4.5 Data production files

A data production file (or production request) concisely defines what processing steps have been performed on an LHCb dataset. Each file can contain multiple steps, as for example, a Monte Carlo production consists of up to eight steps. In the reconstruction or stripping production files, there is typically only one step. Each of these processing steps may require one or several input files and produce one or several output files (such as data files and log files).

Example 4.5.1. An example of a production file is shown in Figure 4.4. This file represents the reconstruction step in the real data production of 2016, where the BRUNEL V50R3 application was used. The number of events (labelled as "Number of events") indicates -1, which means that all available raw events were processed. Other features of the file will be discussed in Section 4.6.1. We can observe that even though the current production files resemble dictionaries, they are not properly formatted and hence cannot be easily computationally handled. They are created for the web representation and hence contain special characters and formatting.

Data production files are a necessary component to reproduce the LHCb data production (discussed in the Chapter 5), hence it is essential that they are fully preserved. However, they depend on the LHCb software (introduced in Chapter 3), which also needs to be curated for the future. Therefore, information about the LHCb software should also be stored in the provenance database. This information should include the software tag of each release (name and version) and links to their dependencies.

The Monte Carlo production files are created using *models*, which are templates that describe the structure and most of the workflow configurations. Typically, the LHCb physicists send requests for the Monte Carlo simulation sample, where they specify the missing configurations in the template, such as the number of events and the event type. The event type is defined by a decay file (DEC FILE) that describes a particle decay, which is then added to the DecFiles database package utilised by GAUSS to configure the generation of the new events. There are currently approximately sixty models, which were the predecessors of over nine thousand Monte Carlo production files. Due to their similarity to the production files, they need to be preserved in the same manner.

4.6 Database implementation

The LHCb data provenance needs to be systematically stored in a database. There are various types of databases, and for our implementation, I explore two different approaches which are relational and graph databases.

Relational databases have been an essential component of software applications since the 1980s. They store highly structured data in tables with predetermined columns of specific types and many rows of the same type of information. Therefore, the data needed to be transformed into required type and strictly structured to follow a database schema before it could be used in the applications.

In relational databases, references to other rows and tables are indicated by referring to their indexes, known as primary and foreign key attributes. The primary key represents one or more columns whose data is used to uniquely identify a row in a table, while a foreign key is one or more columns in the table that refers to the primary keys in other tables. Joins are computed at query time by matching primary and foreign keys of the many rows of the joined tables. These operations are computationally and memory-intensive and their complexity grows exponentially with the number of joins.

Example 4.6.1. To illustrate this approach, we consider an example of a relational database to document the LHCb data production as follows. This can be implemented in various ways depending on the desired granularity of the data. In this case, two tables represent the list of software applications and the list of processing passes shown in Table 4.1. In the data production table, the *Software ID* indicates what application was used in each production. The complexity of joining the two tables to obtain the matching would be $O(N \cdot M)$, where M and N are lengths of the tables respectively. Considering that there are over twelve thousand data productions at LHCb, this approach is not optimal.

Software name	ID	Data production	Software ID
DaVinci v32r2p1	012	COLLISION12 Stripping20	012
Brunel v48r2	013	COLLISION12HL Stripping20	012
		COLLISION15 Reco15a	013

Table 4.1: An example of tables in relational database

Alternatively, there is a newly emerging technology of graph databases. They enable assembling the implicit connections ("relationships") between the objects, which are indicated by foreign keys in the relational databases. Each object ("node") in the graph database directly contains a list of relationship records that represent its connections to other nodes. These relationships are organised by type and direction and may have additional attributes. In computationally intensive operations such as graph traversals, the database uses these lists. It has direct access to the connected nodes, eliminating the need for a complex search and match computation. Therefore, performing such operations would take substantially less time than in relational databases. For instance, querying relational databases (when using *join* operators) may take hours or days, while querying graph databases only milliseconds to minutes, but generally not longer.

One of the most prominent graph databases and the technology used in our solution is NEO4J [136]. It is an open source NoSQL graph database implemented in Java.

Objects in NEO4J are managed with the SQL-inspired and declarative query language called CYPHER. CYPHER is built on the basic concepts and clauses of SQL but has a lot of additional graph-specific functionalities, making it convenient to work with. It was initially created by Neo Technology for the graph database NEO4J. Since then, it was published through the openNeo4j project and adopted by several other graph database vendors.

To implement the LHCb provenance database we relied on NEO4J and CYPHER. The

main components of the database are *Production* nodes to document the LHCb production data flow, *Project* to document the LHCb software stack and *Platform* to document the hardware and runtime environment. Together the nodes represent the provenance of the LHCb datasets.

4.6.1 Production

A production node captures a list of steps that defines one LHCb data production workflow. These nodes contain production files as shown in Section 4.5 and additional derived features.

There are two types of *Production* nodes, and those are the simulation and real data productions. Furthermore, there is a subset in the Monte Carlo production files which represents the Monte Carlo models. The models are templates for the production files as introduced in Section 4.5.

Example 4.6.2. Every node is described with a list of attributes in JSON as presented in Figure 4.5.[3] The `Name` value includes the year of data-taking (here 2016), the beam energy and the kind of collision, in this case, that is *Proton-Argon*. `ID` stands for the identification number of the production and `DQflag` is a data quality flag. Every step has its own `ID` number and a list of configurations, such as `DDDB` and `CondDB` tags (which define the detector and the conditions respectively), python application configuration files (marked as `Options`) and the `Input` and `Output` data formats.

To populate the Production nodes, we use information from the DIRAC production files. Currently, the productions files are stored in a file format that is not easily usable and as such not advisable for preservation. They contain a number of special characters and nested dictionaries, thus can be properly interpreted only through a web browser (shown in Figure 4.4). This is why the production files were rewritten and restructured to be stored in a simple dictionary.

Furthermore, we observed that linking data productions and physics data analyses is difficult because there is no easily accessible connection link from a dataset to its native production file. Using the information from the dictionary, we reconstructed the bookkeeping path for each production node.

Example 4.6.3. By joining together the values of the fields `"SimCondition"`, `"ProPath"` and `"EventType"`, we created a bookkeeping path for each production node. For example, if the fields in the dictionary are as follows:

[3] JSON (short for JavaScript Object Notation) is a file format that uses human-readable text to transmit data objects consisting of key–value pairs and array data types [137].

```
{
  "ID": "31034",
  "Name": "Reco16-Beam2510GeV-MagDown-ProtonArgon",
  "Type": "Reconstruction",
  "EventType": "90000000 Full stream",
  "ProcessingPass": "Real Data",
  "DQflag": "OK",
  "Step": [
    {
      "ID": "129609",
      "Application": "Brunel-v50r1",
      "Options": "\$APPCONFIGOPTS/Brunel/DataType...",
      "DDDB": "dddb-20150724",
      "CondDB": "cond-20160517",
      "Extra": "AppConfig.v3r272",
      "Input": "RAW(Y)",
      "Output": "BRUNELHIST(Y),FULL.DST(Y)"
    },
    { ... }
  ]
}
```

Figure 4.5: An example of a JSON dictionary describing the LHCb data production. The figure shows only the first step of the production workflow.

```
{
    "ProPath":"Sim05a/Trig0x40760037Flagged/Reco12a/
        Stripping17NoPrescalingFiltered",
    "SimCondition":"Beam3500GeV-2011-MagDown-Nu2-EmNoCuts",
    "EventType":"11114001",
},
```

the bookkeeping path is:

```
MC/2011/Beam3500GeV-2011-MagDown-Nu2-EmNoCuts/Sim05a/
Trig0x40760037Flagged/Reco12a/Stripping17NoPrescalingFiltered/
11114001.
```

The Production nodes, after creation, are linked to the LHCb software nodes.

4.6.2 Project

The *Project* node describes one project of the LHCb software stack.[4] It is defined with a project name and version, for example, BRUNEL V44R8 or DAVINCI V36R2. The links

[4]The LHCb projects are sometimes referred to as *modules*.

Project	Platform
DAVINCI v38r0	`x86_64-slc6-gcc49-opt`
	`x86_64-slc6-gcc48-opt`
	`x86_64-slc6-gcc48-do0`
	`x86_64-slc6-gcc49-dbg`
	`x86_64-slc6-gcc48-dbg`

Table 4.2: An example of a project and its compatible platforms.

in between the projects are shown in Figure 4.6 and the links between data productions and projects are shown in Figure 4.7.

To populate the Project nodes, we used information from the Software Configuration database [138], where the links between the projects of the software stack were harvested directly from Gitlab (and previously from SVN). Finally, each project has a dependency on hardware, which is described with the node *Platform*.

4.6.3 Platform

The *Platform* node captures a computing environment that support the LHCb software. It is the simplest node of the database with only one property, which contains a set of compilation tools and flags used to compile the software.

Example 4.6.4. For instance, the node `platform: x86_64-slc6-gcc49-opt` represents a hardware with optimised (opt) `x86_64` architecture (in 64-bit mode) of SCIENTIFIC LINUX CERN 6. Conventionally gcc49 stands for GNU Compiler Collection (GCC) version 4.9.

The edges between the *Projects* and the *Platforms* define the compatible hardware platforms for each project. A list of compatible platforms for DAVINCI V38R0 is shown in the Table 4.2. The platforms labelled with "dbg" and "do0" are used for debugging. This means that GCC options are set to be `-Og` and `-Oo`, which are the options that allow partially and fully verbose debugging output respectively. For data production purposes optimised platforms labelled as `opt` are used.

In conclusion, the provenance database records the data provenance by explicitly linking the dataset metadata to the processing sequence they went through. It mostly follows the formal PROV structure as described in Section 4.1, except for the absence of the executor, as the activities are organised by the LHCb collaboration.

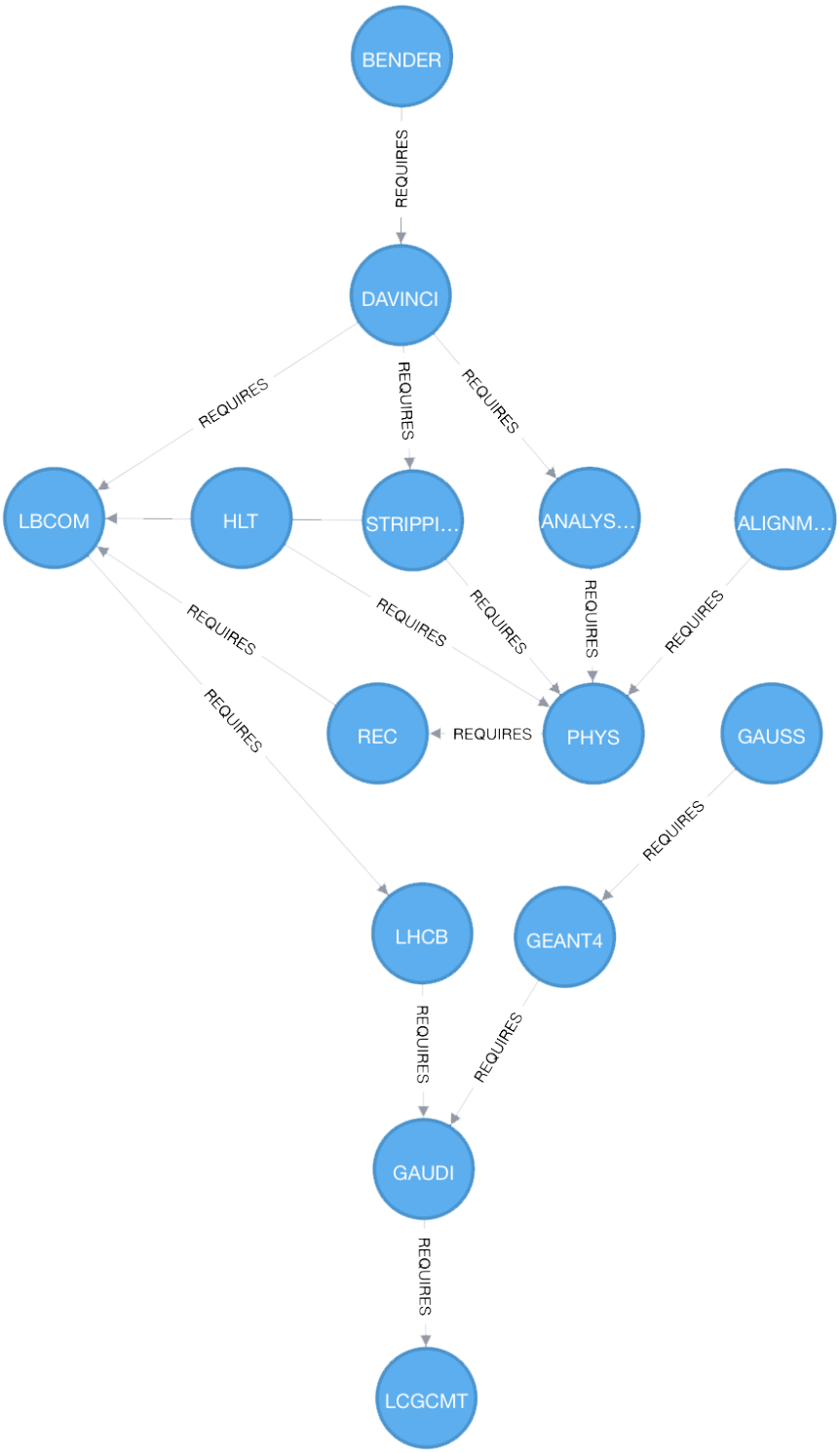


Figure 4.6: An illustration of software dependencies in the graph database. The image shows the structure of the LHCb software stack and the relationships between the projects. For clarity not all possible relationships are shown in the figure.

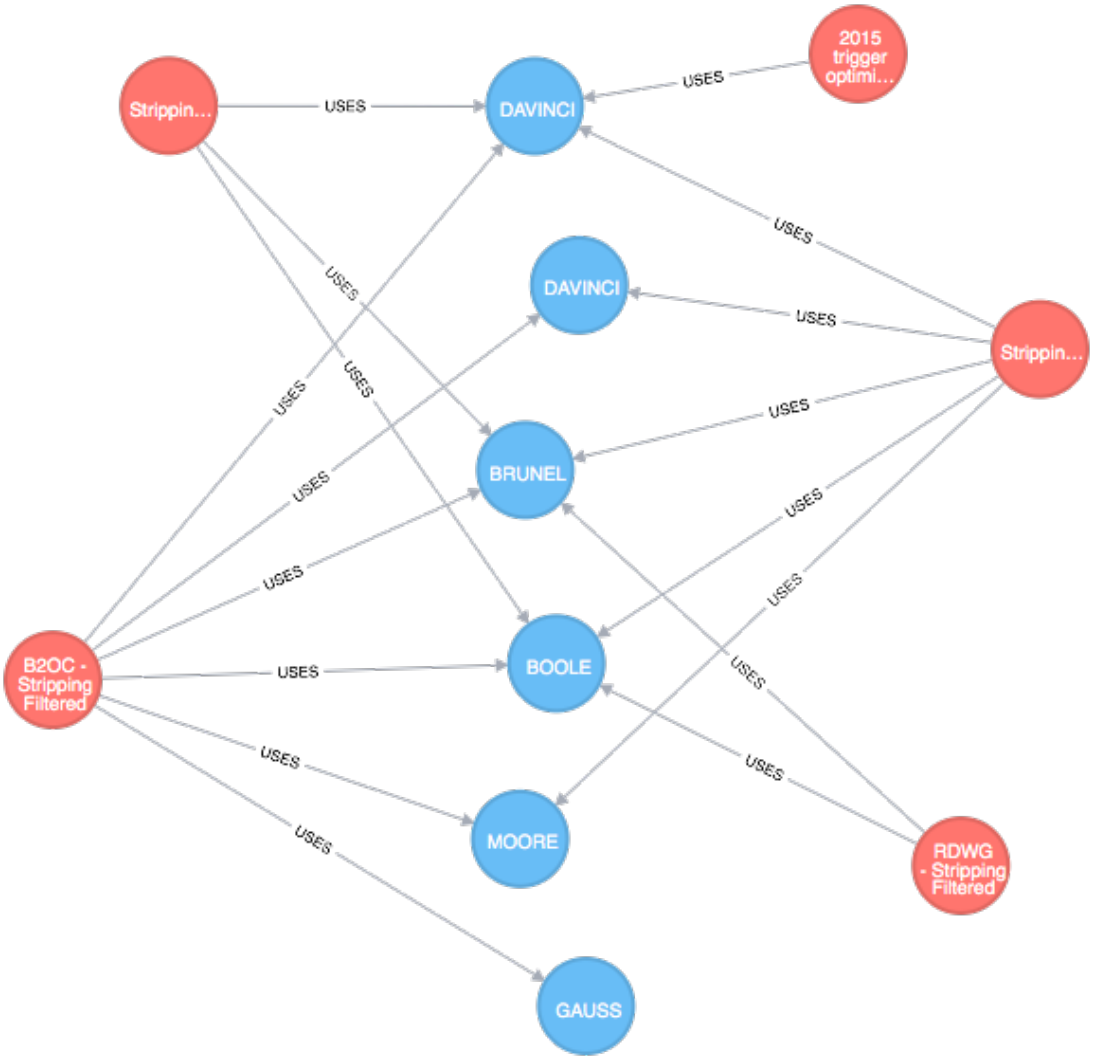
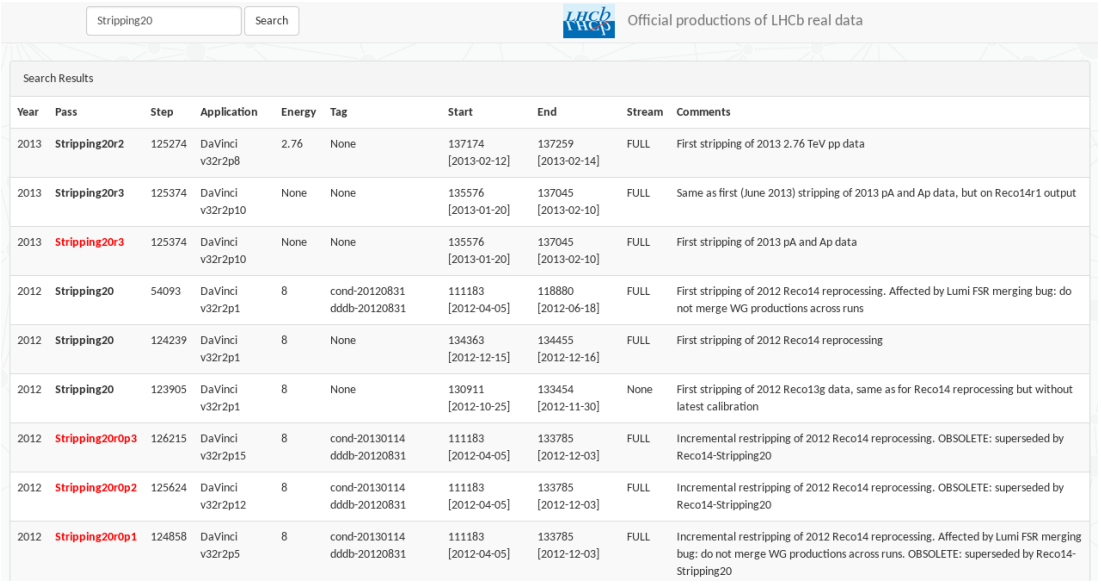


Figure 4.7: Appearance of the nodes in the database. The blue nodes are *Projects*, and the red nodes are *Productions* (in this case Monte Carlo production). All productions use one of the projects in their workflows. For clarity not all possible relationships are shown in the figure.



Year	Pass	Step	Application	Energy	Tag	Start	End	Stream	Comments
2013	**Stripping20r2**	125274	DaVinci v32r2p8	2.76	None	137174 [2013-02-12]	137259 [2013-02-14]	FULL	First stripping of 2013 2.76 TeV pp data
2013	**Stripping20r3**	125374	DaVinci v32r2p10	None	None	135576 [2013-01-20]	137045 [2013-02-10]	FULL	Same as first (June 2013) stripping of 2013 pA and Ap data, but on Reco14r1 output
2013	**Stripping20r3**	125374	DaVinci v32r2p10	None	None	135576 [2013-01-20]	137045 [2013-02-10]	FULL	First stripping of 2013 pA and Ap data
2012	**Stripping20**	54093	DaVinci v32r2p1	8	cond-20120831 dddb-20120831	111183 [2012-04-05]	118880 [2012-06-18]	FULL	First stripping of 2012 Reco14 reprocessing. Affected by Lumi FSR merging bug: do not merge WG productions across runs
2012	**Stripping20**	124239	DaVinci v32r2p1	8	None	134363 [2012-12-15]	134455 [2012-12-16]	FULL	First stripping of 2012 Reco14 reprocessing
2012	**Stripping20**	123905	DaVinci v32r2p1	8	None	130911 [2012-10-25]	133454 [2012-11-30]	None	First stripping of 2012 Reco13g data, same as for Reco14 reprocessing but without latest calibration
2012	**Stripping20r0p3**	126215	DaVinci v32r2p15	8	cond-20130114 dddb-20120831	111183 [2012-04-05]	133785 [2012-12-03]	FULL	Incremental restripping of 2012 Reco14 reprocessing. OBSOLETE: superseded by Reco14-Stripping20
2012	**Stripping20r0p2**	125624	DaVinci v32r2p12	8	cond-20130114 dddb-20120831	111183 [2012-04-05]	133785 [2012-12-03]	FULL	Incremental restripping of 2012 Reco14 reprocessing. OBSOLETE: superseded by Reco14-Stripping20
2012	**Stripping20r0p1**	124858	DaVinci v32r2p5	8	cond-20130114 dddb-20120831	111183 [2012-04-05]	133785 [2012-12-03]	FULL	Incremental restripping of 2012 Reco14 reprocessing. Affected by Lumi FSR merging bug: do not merge WG productions across runs. OBSOLETE: superseded by Reco14-Stripping20

Figure 4.8: Presentation of the provenance database on the web. Obsolete processing passes are marked in red.

In order to make this information accessible and usable, we implemented an application programming interface (API) and a webpage to present the database.

4.7 Database API

The data from the NEO4J database is presented on a web server implemented in Flask [139], as shown in Figure 4.8. Flask is a micro web framework which allows rapid development of web applications. The metadata is presented in a descending list that marks recommended and obsolete stripping versions. In addition, there is a search engine that allows browsing through this metadata and the software dependencies.

Since it is of particular interest for the preservation efforts that the database can be efficiently accessed, we created an application programming interface (API) to handle this communication. The API defines a set of functions that query the database and provides responses. It is implemented in PY2NEO [140], which is a client library and toolkit for working with NEO4J from within the python applications and the command line. We use the Representational State Transfer (REST) [141] architectural style to create a request/response mechanism between the server and the client. Some of the basic functionalities are listed as follows:

- `createProject(name, version)` Creates a new instance of a project.
- `createPlatform(platform)` Creates a new instance of a platform.

- `createProjectPlatformRelationship(p, v, plat)` Creates a relationship between a project *p* and a platform *plat*.
- `createProjectProjectRelationship(p1, v1, p2, v2)` Creates a relationship between a project *p1* and a project *p2*.
- `getProduction(ID)` Returns a *Production* entity with a given *ID* number.
- `listProjects()` Returns the list of *Projects*.
- `listPlatforms(Project)` Lists the suitable *Platforms* for a given *Project*.
- `listRequirements(Project)` Lists the required project dependencies for a given *Project*.

Regarding analysis preservation, the analysts are encouraged to document their work by specifying data provenance together with the analysis code and methods. Data provenance was not readily accessible to the analysts, which imposed an obstacle in preservation. By linking the bookkeeping paths to the production metadata within the graph database, this process is made significantly more accessible. With the API, we are able to automate the process of recording data provenance for analysis preservation. One of the potential uses of the API is the CERN Analysis Preservation portal [142], which is discussed in Chapter 8.

4.8 Graph mining

The database consists of thousands of nodes connected in various ways, which allow us to draw valuable conclusions by studying these connections. The process of finding and extracting concealed information from graphs is called graph mining. Currently, it is not completely clear what information is worthwhile for the preservation purposes as the challenges that may arise in the future cannot be perfectly anticipated. Here we explore several scenarios in which graph mining of the provenance database can aid data recovery and reuse.

The first scenario covers learning about the software stack by traversing the graph. One of the major assets of the LHCb software stack is that each project is built depending on another. Each project version requires a specific set of other projects to run, and this database allows us to understand and employ these dependencies.

Example 4.8.1. To demonstrate graph traversal using the CYPHER query language, we search for the path ***p*** from an application at the top level of the LHCb software stack to the base level.

In this example, the top level application is DAVINCI V33R1, and the base of the stack is the GAUDI framework. The result of this query is a path which normally contains several nodes and includes all the projects that are required to run the top level application DAVINCI V33R1.

```
MATCH p = (a:Project{name:'DAVINCI'})-
    [r:REQUIRES*..]->
    (d:Project{name:'GAUDI'})
    RETURN p LIMIT 1
```

The output of the previous query is:

DAVINCI v33r1 → ANALYSIS v10r3 → PHYS v16r3 → REC v14r3
→ LHCB v35r3 → GAUDI v23r5

The second scenario could be looking for all the datasets affected by a corrupt version of a project. We can perform troubleshooting of the software stack by traversing the graph to identify the datasets handled by a faulty software component.

Example 4.8.2. To learn how many productions have been created using the project PHYS V18R2P1, we run the following command. The query returns a count of production nodes linked by the *requires* or *uses* edges to the node PHYS. In this case, the link between the project and the productions is indirect, as shown in Figure 4.9 .

```
MATCH (n:Production)-
    [:REQUIRES|:USES*1..100]->
    (b:Project{name:"PHYS", version:"v19r2p1"})
RETURN COUNT(n)
```

The query returns an answer of 280, meaning that there are 280 productions created with the project PHYS V18R2P1. If there was an error in the software that may have affected the data, a list of productions influenced by the error could be derived, and the collaboration's access to these datasets could be restricted before the error is solved.

Furthermore, using clustering, we can classify the Project nodes into groups where each of them is prerequisite for a specific subset of the data (e.g. 2011 data). This is particularly important for testing the projects against their corresponding datasets and verifying that they work in the same way in the future as they work today.

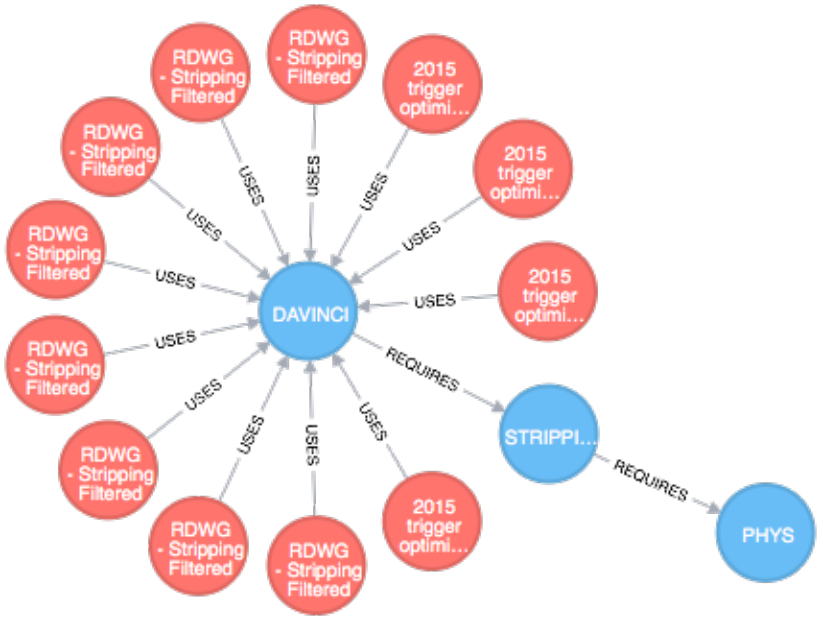


Figure 4.9: View in NEO4J database that shows paths from production metadata to PHYS V19R2P1.

Example 4.8.3. Consider a use case where it is necessary to process a specific subset of the data, for example, *Collision 16* (stands for data created in 2016). Using the graph database, the set of required applications for this task is easily discoverable. The CYPHER query below demonstrates how to obtain a unique set of application versions for the trigger software MOORE to process the data of 2016.

```
MATCH (a:Production {year:"2016"})--(p:Project {name:"MOORE"})
RETURN COLLECT (DISTINCT p.version);
```

The result of the query is shown below.

Moore versions
v25r4
v25r5
v25r3p1

Finally, it is straightforward to identify the most used software versions across the LHCb timeline. These studies can be used to declare a "legacy" version of each project for a given time period. Such version is then used for the future reconstructions of the old data, to take advantage of the improved selection algorithms.

Example 4.8.4. The Project nodes with the highest degree in the graph represent the most used versions. The CYPHER code to find a list of such nodes is shown below. In this example, both the simulation and real data productions are included, and the reconstruction and stripping information dates from October 7, 2009, to November 28, 2016. [5]

```
MATCH (p:Project)-->(a:Production)
RETURN id(p), count(*) as degree
ORDER BY degree
```

Table 4.3 shows results of the query for the projects: GAUSS, BOOLE, MOORE, LHCB, BRUNEL and DAVINCI. The most used project versions for simulation production are GAUSS V45R7, BOOLE V26R3 and MOORE V14R8P1. The most used version of LHCb is LHCB V35R4. BRUNEL V43R2 is by far the most used project version for reconstruction. It is not a surprise because it was used for reconstruction of all Run 1 data of 2010, 2011 and 2012. Furthermore, it was used in reconstructions for proton-ion and ion-proton collisions in 2013, for *pp* collisions in 2010, for proton-ion collisions in 2012 and for reprocessing of *pp* collisions in 2012 and 2013. This resulted in BRUNEL V43R2 appearing in two different "patch versions" in almost 2000 production files.

Similarly to BRUNEL V43R2 the most frequent DAVINCI version was used exclusively on Run 1 data in 2010, 2011 and 2012. It is DAVINCI V32R2, and it was widely used after the reconstruction with BRUNEL V43R2 in *Reco14-Stripping20* processing passes. These most frequently used applications are possibly the most representative ones for a particular data-taking period, and they could be released open access, for example in a DOCKER container, and provided with the LHCb open data to the general public.

Similarly, we could find the projects with the lowest degrees, which represent the least used project versions. Looking for the least used project versions can be used for identifying obsolete projects, which should be archived and potentially removed from CVMFS. However, a further investigation would be needed, since these projects might have been used for special runs at LHCb, meaning that they would still be valuable for preservation.

4.9 Chapter summary

In this chapter, I presented the design and implementation of the graph database to keep the provenance of the LHCb data. Even though this is a fully working implementation,

[5] The simulation production files were not always labelled with a time stamp.

Application	Frequency	Application	Frequency
Gauss v45r7	1019	Boole v26r3	4552
Gauss v45r3	904	Boole v23r1	763
Gauss v45r10p1	645	Boole v30r1	702
Gauss v45r9	568	Boole v21r9	320
Gauss v49r1	464	Boole v24r0	181
Application	**Frequency**	**Application**	**Frequency**
Moore v14r8p1	3375	LHCb v35r4	2029
Moore v12r8g3	1556	LHCb v33r1	723
Moore v12r8g1	752	LHCb v31r7	320
Moore v24r2	592	LHCb v35r1	146
Moore v20r4	577	LHCb v38r6	128
Application	**Frequency**	**Application**	**Frequency**
Brunel v43r2p11	1390	DaVinci v32r2p1	2606
Brunel v41r1p1	755	DaVinci v32r2p3	1184
Brunel v48r2	431	DaVinci v29r1p1	702
Brunel v43r2p10	406	DaVinci v36r1p1	489
Brunel v37r8p5	320	DaVinci v40r1p3	325

Table 4.3: List of the most used project versions by November 2016.

it is a prototype, and it needs additional development before it can be actively used by the community. I explored several scenarios how the database can aid in extracting information about the LHCb software stack. However, the main feature of the database is that it provides complete information on how to recreate the LHCb data production workflows. I investigate how this can be achieved in the following chapter.

Chapter 5

Data processing on the cloud

Enabling data analysis and data processing outside of the CERN architecture is useful for several reasons. First, it benefits the portability of physics analyses, which then can be executed at the number of different locations or infrastructures. Second, it enables scholars outside of the collaboration to run experimental software on (open) data. This would be useful for both citizen science and educational purposes. Finally, once the current LHC experiments become obsolete, the main LHC computing infrastructure may be used for future CERN experiments. Hence, for long-term preservation purposes, alternative solutions for the LHC data processing need to be identified. In this chapter, we explore the potential and flexibility found in cloud computing.

In this chapter, I present a methodology to execute the LHCb data production on the cloud, whilst preserving the original software and hardware dependencies. Finally, I compare the events obtained on the cloud to the official LHCb events to evaluate the new methodology.

5.1 Cloud computing services for data preservation

Raw experimental datasets are stored in multiple copies on the Grid for data redundancy reasons. However, regarding the storage of derived and obsolete datasets, it is not obvious whether it is beneficial to save them or erase them.[1] Such decisions should be made based on the cost of saving versus the cost of reproducing them, and also taking into account the probability that these datasets will be used in the future. It is likely that the cost-effective strategy is to be able to recreate or reprocess a dataset rather than to keep everything ever created in storage. Here we explore the first attempts to reproduce the data processing outside of the LHC computing

[1] As discussed in Chapter 3 all legacy reconstruction and stripping datasets will be preserved.

Grid.

Cloud computing presents a cutting-edge practice of using a network of remote computers to store, manage and process data. It provides an effective, on-demand and scalable allocation of computing resources [143], which is suitable for processing large amounts of data. According to the current trends [144], the cost of the infrastructure, the storage space and its maintenance is going to be significantly lower in the future, which is desirable for the preservation efforts.

Our goal is to mimic the WLCG within a cloud computing infrastructure and to process the LHCb data in an identical manner as it would be processed on the WLCG. In order to achieve this, we identify required components and methods. To manage the resources on the cloud, we use OpenStack [145], which is the open source cloud computing software deployed on CERN's private cloud. The cloud is provisioned for the use at CERN, however, it works in the same fundamental way as other commercial clouds. The LHCb collaboration, together with the other experiments, have allocated resources on the CERN cloud to be used for preservation purposes.

5.2 Technical requirements

The LHCb software is developed for the official CERN operating systems, which currently include SCIENTIFIC LINUX CERN 6 and CERN CENTOS 7. Previously SCIENTIFIC LINUX CERN 5 was also used, but it is now obsolete. In order to reproduce a data production workflow, we should use the same operating systems that were once used. However, there is no guarantee that this would be possible with new hardware in the future. This is why we use virtualisation technologies, which are invaluable for capturing software and their system dependencies.

5.2.1 Virtual Machine

A virtual machine (VM) is an emulation of a computer system, which can support and run different runtime environments. They are used in a number of scenarios like for example for software testing and resource utilisation purposes, and due to their capability to capture and encapsulate software, they have also found their use in scientific preservation and reproducibility. Other important benefits include portability, manageability, and security of encapsulated resources.

In order to create a VM, it is necessary to have a hypervisor, or VM monitor, which is a software layer between the host machine and the VM (as shown in Figure 5.1a). It interacts with the hardware and provides an interface to share the available resources with the guest operating systems (OS). The hypervisor is also responsible for creating a virtual hardware and software environment to run and manage virtual machines. On top of the hypervisor, a VM runs a full copy of a guest OS (which can be for example Linux, Windows, OS X) and a virtual copy of the hardware that the OS needs to run [146]. Applications that are deployed with VMs are isolated on the guest OS, where they have access to binaries and libraries located within it.

The advantage of using virtual machines is that they provide environments that are completely isolated from each other and from the host OS. They also allow full virtualisation, meaning that each VM can, for example, have its own CPU virtualisation. This is particularly significant for running obsolete computing infrastructures. The drawback of using VMs is that they quickly can allocate a lot of RAM and CPU cycles, since they each need an entire guest operating system. In addition, there is a decline in performance as they do not have a direct access to the hardware, but instead the interaction with the hardware happens through a number of software layers.

The CERN Virtual Machine has been created to provide a uniform and portable environment for high-energy physics applications. New developments have now allowed for these virtual machines to be unprecedentedly small and thus more convenient to use. The size of Micro-CernVM [147] is only about 12 MB, and it can be used to on-demand load other CERN operating systems through CERN Virtual Machine File System (CVMFS), which was introduced in Chapter 3.3. For instance, in order to boot SCIENTIFIC LINUX 6, Micro-CernVM would use an additional 100 MB of storage space.

5.2.2 Virtual containers

Virtual containers represent a novel virtualisation technology pioneered by Docker Inc., which appeared in 2014. It provides lightweight virtual environments on various host operating systems. According to Docker Inc., the definition is:

> ”DOCKER containers wrap a piece of software in a complete filesystem that contains everything needed to run: code, runtime, system tools, system libraries, anything that can be installed on a server. This guarantees that the software will always run the same, regardless of its environment.“ [148]

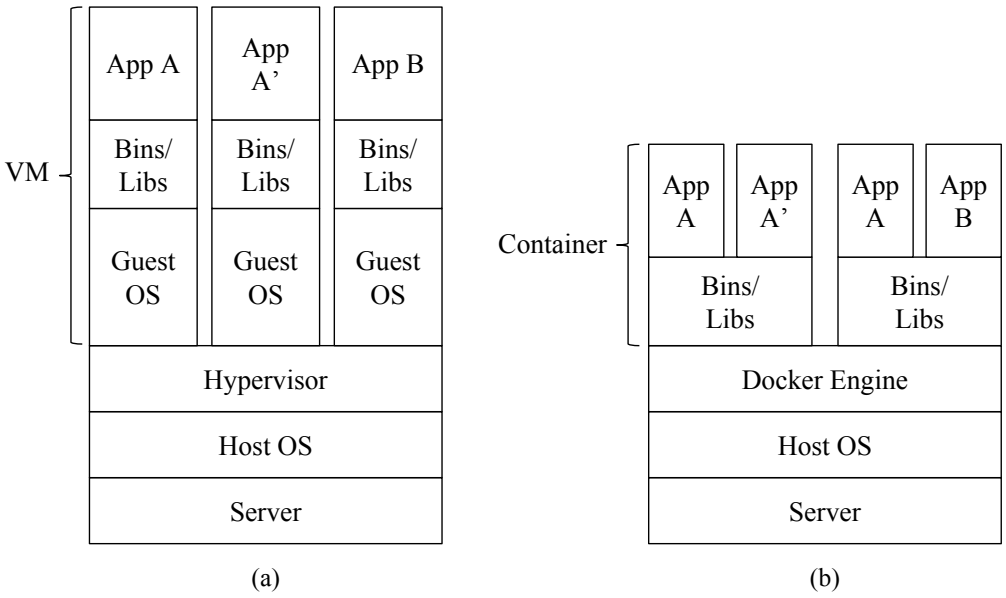


Figure 5.1: The difference in structure between virtual machines and DOCKER containers on a host computer.

DOCKER containers are created in a way that allows a high level of runtime environment customisation. A DOCKER image is created with a Dockerfile, which has a simple syntax for defining the steps necessary to create and run the image. The first line of the Dockerfile defines a base image, and each new instruction makes modifications to the system environment (e.g. installing new software), which makes recreating and modifying images fast and lightweight.

Virtual containers represent isolated processes that share system resources with the host machine, as shown in Figure 5.1b. This means that the applications deployed on DOCKER containers share the operating system and, where possible, system binaries and libraries. The containers do not virtualise the hardware and do not typically employ a guest operating system. The DOCKER engine acts like the aforementioned hypervisor, and it is the component that runs the containers.

One of the biggest differences between the VM and DOCKER container technologies is that the containers can share system resources with other containers on the same host machine. This allows for a higher performance than VMs since there is no guest OS for each container. Other advantages of using virtual containers are that they are typically smaller in volume (memory use), which means that they often can be faster. However, their drawback is that they cannot support a scenario where each application requires a specific OS, which is easier to achieve with VMs. There are also potential security issues since the containers run processes that have direct access to the host machine processes and memory. Hence, it is not recommended

to run a container as a superuser.

DOCKER containers have already found their role at LHCb as they are actively used for the software nightly builds, to provide infrastructure standardisation.[2] The LHCb nightly build system compiles and tests the LHCb software on all supported operating systems, using a uniform cluster of CERN CENTOS 7 machines. In addition, DOCKER is used to provide a standardised runtime environment and starting point for users at various LHCb workshops.

DOCKER images can be published to and retrieved from a DOCKER global registry called DOCKERHUB [149], which also allows browsing through the published images, both official and custom-made. This is particularly important to the LHCb data preservation efforts, since this registry could allow versioning, preservation and sharing of the LHCb software and physics analyses.

5.3 Implementation

We have seen in the previous chapter that data production consists of many processing steps, each of which can be fairly complex. Different applications are typically used in each step and output from one stage is used as input to the next (as shown in Figure 5.2).

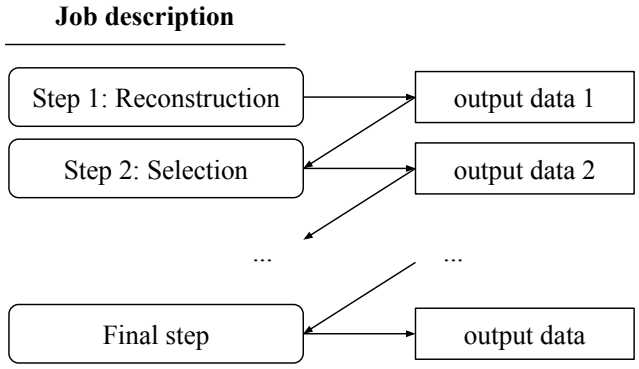


Figure 5.2: A scheme of the LHCb data processing workflow.

For our implementation, we use a combination of a VM and a DOCKER image. The virtual image is based on CERN CENTOS 7, and the base DOCKER image is the same as the one used for the LHCb nightly builds. Access to the LHCb software from the VM and the container is provided from CVMFS. As noted in Chapter 3.3, all LHC software in its final production version can be found in CVMFS [120]. Finally, the steps of data productions are

[2] Infrastructure standardisation enables a cluster of machines that differ in hardware and software to appear homogenous.

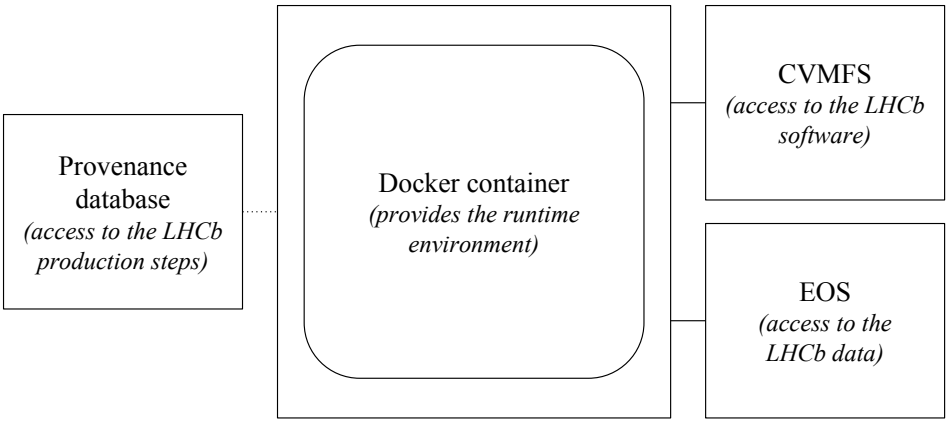


Figure 5.3: A schema of the technical requirements for running the LHCb data production on the cloud.

stored in JSON dictionaries in the LHCb provenance database (introduced in Chapter 4). In order to execute the steps, the dictionary is transformed into a python file which runs the LHCb software mounted from CVMFS. Using these solutions, presented in Figure 5.3, we are able to reproduce the computational environment.

For real data production, raw experimental data is needed as input to execute the production steps. The data can be read from the runtime environment directly from the CERN storage system EOS. However, it is not required for the execution of the Monte Carlo production, as simulation does not require any input data. In the following, the execution of a Monte Carlo production is explained, as its processing workflow in addition to the simulation steps includes the reconstruction and the preselection stages typical for the LHCb experimental data.

For this study, we chose an arbitrary Monte Carlo production to be recreated on the cloud. The production file that defines the workflow steps, creates a simulation sample of the decay $B^0_{(s)} \to \mu^+\mu^-$ with experiment conditions from 2012 [3].[3]

The production workflow comprises five processing steps that are executed one after the other, as shown in Figure 5.4. The processing steps require SCIENTIFIC LINUX 5, which was provided via DOCKER and CVMFS . Using the now obsolete operating system was a good test for this methodology and the virtualisation tools.

Every processing step produces a set of output files, which include job summary files and catalogues that describe the runtime and application metadata, in addition to the output dataset.

The produced events are shown in the event display in Figure 5.5. The figure shows

[3]In the DIRAC system the identification number of the production request is ID 32089.

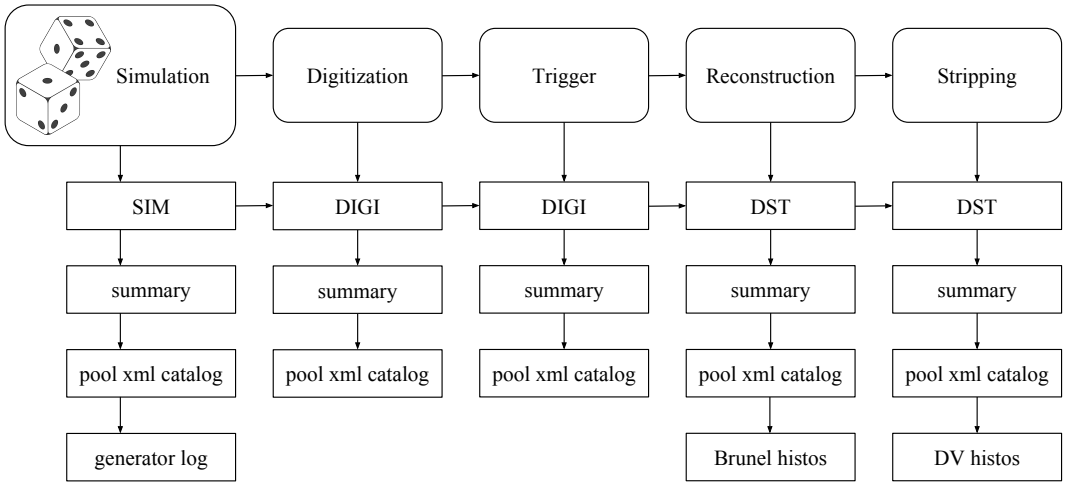


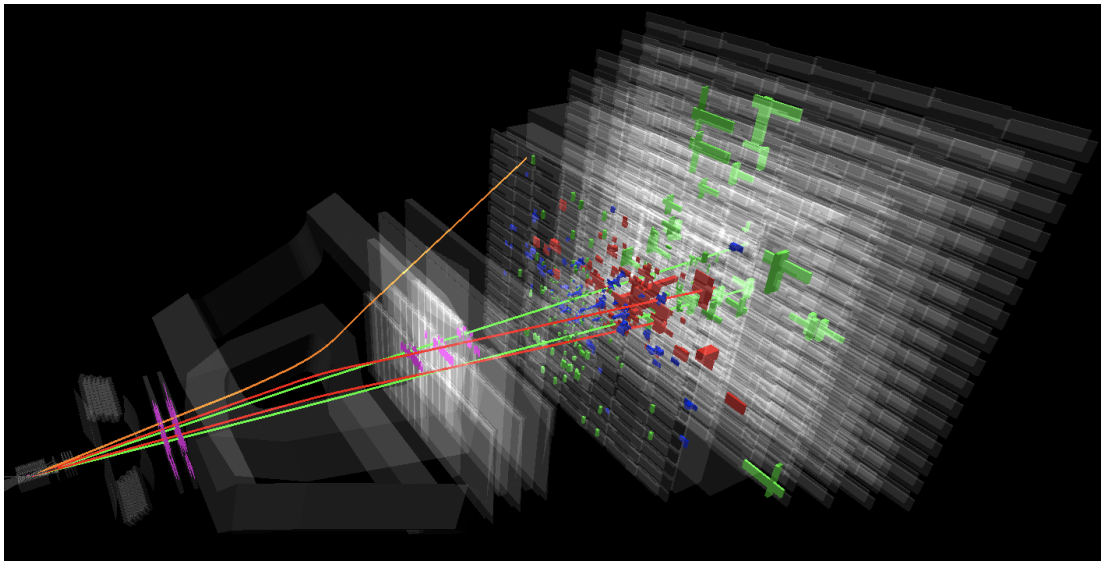
Figure 5.4: A loose schema of the Monte Carlo production stages execution with output files.

two different perspectives of the same event, and the LHCb detector (introduced in Chapter 2). Since these particular Monte Carlo events were used for $B^0_{(s)} \to \mu^+\mu^-$ searches at LHCb [89], the figure features two muons coloured in green among the other particles. Finally, in order to evaluate whether this is a suitable preservation strategy, we compare these events to the official ones created on the Grid by LHCb.

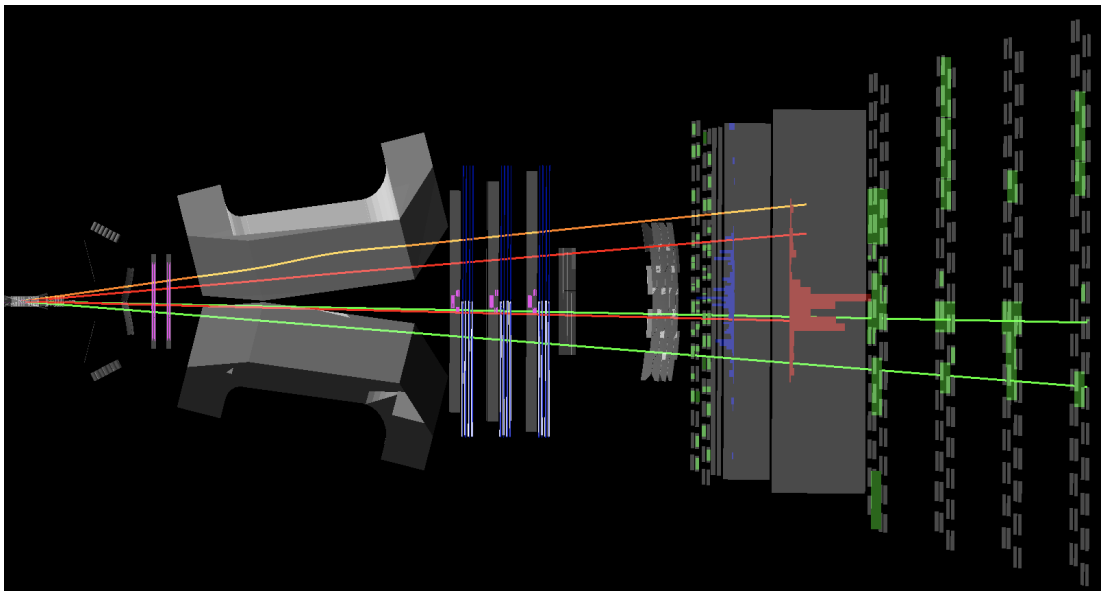
5.4 Result evaluation

Two Monte Carlo samples are equivalent if they are created with the same random seed that uniquely defines the simulated event. Therefore, the cloud created events need to be compared to the events created on the Grid that have the same random seed. In the LHCb Monte Carlo production, the random seed is defined by the `RunNumber` and the `EventNumber` values in the production files, which were introduced in Chapter 4.5. These values are specified in a python application configuration file that produces the MC sample, but they can also be found in the final dataset using the LHCb software or the ROOT framework. For the evaluation of our results, we use the DAVINCI application to explore and compare the content of the files. In the Monte Carlo production, the intermediate files are not saved, therefore we can only compare the final DST files [4].

Inside the DST file, a single event is described with a lot of different data objects (particle tracks, vertices, etc.), which are organised in the Transient Event Store (TES) as introduced in Chapter 3.3.1. These objects are placed in a tree-like structure, and they can be accessed using



(a)



(b)

Figure 5.5: The Monte Carlo events created on the cloud visualised in the LHCb event display. Figures show a different perspective of the same event. Pion track is marked in orange, kaons are marked in red and muons in green. Hits in the inner tracker (IT) are in purple. Hadron calorimeter (HCAL) hits are in red and electromagnetic calorimeter (ECAL) hits are in blue. The hits in the muon system are marked in green.

the LHCb software.

For the comparison of the Monte Carlo events created on the cloud to the official ones, we studied the DST summary records and compared the reconstructed tracks. The events were serialised in the following way: all particle tracks are represented with their p and p_T values, followed by a list of the LHCb ID numbers that represent specific locations of the track hits in the detectors.

Example 5.4.1. Consider the example where the run number and event number are 5678 and 1234 respectively. It contains 104 track segments, which are serialised in the following way:

```
Run number 5678
Event number 1234
track.p 10623.75
track.pt 517.171370169
lhcbIDs:
269011501
269016140
269019756
269024392
269028112
269032821
269036436
269041072
269044684
269662470
269666753
269670642
269674931
269678824
(...)
```

The serialised events were stored in text files that were then compared and found to be identical to each other. Furthermore, the values describing the particles and their tracks were sorted in the exact same order. We conclude that the Monte Carlo production on the cloud can

create bit-wise identical events as the Grid infrastructure and that this methodology can indeed be used for data preservation.

5.5 Distributed deployment and the REANA project

Particle collisions are independent of each other, meaning that physics events in a dataset can be processed in parallel. This feature allows the implementation of subjobs on the Grid, but it also allows distributed data processing on the cloud. By processing the datasets in parallel, we significantly reduce the execution time.

There are various orchestration tools for managing DOCKER containers on the cloud. KUBERNETES [150] is one such solution that is endorsed by the CERN IT department when working with LHC data. KUBERNETES is a free and open-source container orchestration tool designed to automate deploying, scaling, and operating containerised applications. A KUBERNETES cluster is composed of nodes, where each node is a worker machine, which can be a VM or a physical machine. Each node contains software to run containers managed by the KUBERNETES control plane. The control plane represents a set of APIs and software that runs on master nodes and handles interactions with the users.

In order to advance the data analysis preservation on the cloud, a new project called REANA (REusable ANAlysis) was created. The primary goal of the project is to allow automation and transparency in running of the preserved analyses from CAP. Using information about the input data, software, computational environment and computational workflows, REANA allows its users to submit and run analyses on the cloud. REANA is implemented with DOCKER containers on a KUBERNETES cluster hosted at CERN. So far, only simple data analyses have been realised [151]. However, this is a promising project, and by employing the procedure described in this chapter, it can also facilitate the LHCb data production in addition to the physics analyses. Future aspirations include facilitating execution of ongoing data analyses on REANA, even outside of the HEP collaborations.

5.6 Chapter summary

The LHCb data production is traditionally done on the WLCG. However, in this chapter, we explored the possibilities of processing the data on the cloud, outside the Grid infrastructure. The Monte Carlo events simulated on the cloud proved to be identical to the original ones. This means that cloud computing can indeed be used as an alternative to the CERN Grid computing

and that it is a reassuring approach for the data preservation. In addition, as increasing volume of data is released open access, the LHC collaborations should recommend an infrastructure to analyse data, and thus facilitate outreach and citizen science. In this case, such recommendation can be using a methodology explained in this chapter on a public cloud infrastructure, which may be owned by a business, academic or government organisation, but is provisioned for open use by general public [143].

This chapter presented one solution using cloud computing, however, we note that over time computing resources become more efficient and powerful, thus allowing new solutions to emerge in the future.

Chapter 6

Provenance tracking in the LHCb software

In this chapter, I introduce newly implemented functionality in the GAUDI software framework that I have designed and developed to track and capture provenance of a dataset. It is implemented as a service within the framework that traverses through the application configurations to capture and store them as an object inside the output file. Using the captured information, the dataset can be independently reproduced, thus eliminating the need for the original python application configuration file. I explain the implementation of the service in detail and give an example of its application.

6.1 Provenance tracking for physics analyses

In physics analyses, the provenance of data files is essential to understand how the file was produced and what kind of objects, in this case, particles, it describes. This concept was introduced in Chapter 4, where we discussed effective means to capture the origin of the data (in regards to data processing and the LHCb software). However, provenance can also be used to understand and track the series of changes of the derived datasets after data production.

Provenance plays an important role in a number of different scenarios in regards to research preservation and reproducibility. For example, different versions of the same LHCb software application will produce different datasets. In particular, DAVINCI V39R0 and DAVINCI V42R3 are likely to produce slightly different output files even when they are configured in the same way. Therefore, the information about the application needs to be preserved to guarantee

bit-wise reproducibility of a dataset.

In addition, provenance allows an evaluation of correctness of a data file. This can be manifested in a scenario when for example there is a problem with a specific version of the LHCb software. By evaluating that a dataset was not produced with that version, we are more confident of its correctness. On the contrary, a dataset produced with this version of the software should be reproduced with a newer version to reduce errors in data analysis. This is why data provenance facilitates both reproducibility and verification of a dataset.

6.2 ROOT ntuple production

Physics data can be stored in a number of different file formats at LHCb, however, most analyses are performed using data in the ROOT file format [74, 152]. These files are referred to as the ROOT ntuples or just ntuples. A tuple is a finite ordered list of elements, while an ntuple (n-tuple) is a sequence of elements, each of which is described by an n number of attributes. Typically the elements are stored in rows and the attributes in columns. In the case of physics analyses, elements are the particles themselves, and attributes are information such as for example mass and momentum.

A ROOT file is an extremely flexible file format, and it acts like a UNIX file directory, which means that it can store directories (folders) and data objects organised in an arbitrary order [153]. It can store any C++ objects, like for example figures (histograms and plots) and their associated information.

Analysts at LHCb use a wide range of different tools to carry out their physics analyses. However, before they can use custom built (or chosen) tools, they need to extract and retrieve data that is useful for their analysis from the LHCb data streams. This is the final processing step for the LHCb data. It requires the use of the LHCb software, typically DAVINCI, which is why this is possibly the most constrained step of every physics analysis. The analysts submit DAVINCI jobs to the Grid, which select the decay of interest from their stripping lines and potentially apply finer selection on the data. The output files of these jobs are then used for individual physics analysis.

A common practice in physics analyses is to use the latest available (at that time) version of the LHCb software.[1] Using this version is recommended as it captures recent developments that could reduce or solve known issues in the code or perhaps provide new functionality.

[1] This is done by specifying in the command line the keyword `latest`.

By doing this, the analysts do not necessarily recognise or capture what application version they were using at the time, and this information is thus easily lost. When there is a need to reproduce a dataset, the "latest" application version is likely to have been changed in the meantime, which then potentially hinders reproducibility. Another common practice is using the same application configuration file to produce a number of ROOT ntuples with small differences in their configuration. Usually, the output file names indicate how they were produced, but this is often not enough to understand or reproduce them. As these are examples of contextual information that easily gets lost or forgotten, a new provenance tracking service was created within the GAUDI framework to capture the processing information directly inside the ROOT files. This information is comprehensive and can be used to independently reproduce the ntuple.

6.3 The Gaudi framework

The GAUDI framework, as introduced in Chapter 3.3.1, provides an exceptionally flexible development environment for the LHCb software. As previously noted, it supports applications for data collection, reconstruction, preselection, event simulation, visualisation and physics analyses [154]. As a complex software system that contains more than 180,000 lines of code, it is organised in a modular architecture of smaller and more manageable packages.[2] Each component (module) has a well-defined functionality and an interface through which it can interact with the other components. This provides an abstraction to its developers, meaning that they do not need to understand the whole framework to help develop one of the components.

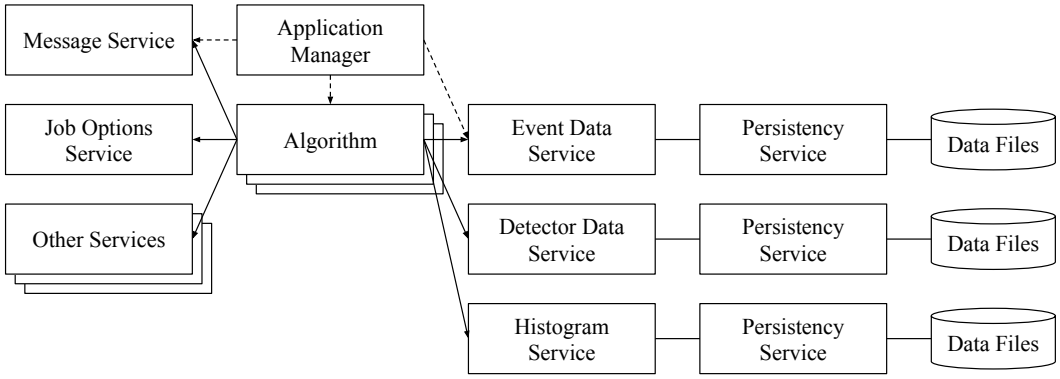


Figure 6.1: Basic GAUDI architecture diagram.

The basic components of the GAUDI framework are shown in a diagram in Figure

[2]The number of lines of code was computed from the command line on the repository cloned from CERN GitLab. In June 2018, it sums up to be 186,577 line of code in `.h`, `.cpp` and `.C` files.

6.1. The diagram represents a hypothetical snapshot of the state of the system showing used components and connections between them.

The Application manager (in code denoted as `ApplicationMgr`) is responsible for controlling the execution of jobs. It creates and initialises the required framework components, and loops over the input data events executing the algorithms. Also, it manages errors and in the end, it terminates the job.

The Algorithms (`Algorithms`), as shown in the diagram, have a central place in the job execution and their primary functionality is to read an input file, process it and produce a new output file. The GAUDI services provide various utilities for the algorithms in the system. The Application manager initialises these services at the beginning of a job, which are then used by the algorithms in the application. Normally, only one instance of each service is used in the job.

The four main services used by the algorithms (and shown in the diagram) are Event data service (`EventDataSvc`), Detector data service (`DetDataSvc`), Histogram service (`HistogramDataSvc`) and Message service (`MessageSvc`). The first three services are used for data access and processing, while the message service provides communication about the progress or errors in the algorithms. The persistency services provide the functionality of saving the output data on the disk. There are also other services in the framework that provide specialised functionalities and that can be enabled and disabled by the users. Each of the services is used by the algorithms via an interface, which is a helper class that defines the functionality of a service through a number of public methods. These methods help the service communicate with the other components of the framework.

The algorithm is configured by a number of parameters. These parameters are specified at runtime via a python application configuration file, which are then managed and applied by the Job Options service (`JobOptionsSvc`). Data Files, also shown in the diagram, represent output data saved on the disk.

6.4 Implementation

The provenance tracking service is named Metadata service (`MetaDataSvc`) [5, 155]. It is created with a purpose to collect information about a job and capture it in an object, which is then saved inside the output file, in this case, the ROOT ntuple. The Metadata service is implemented as a GAUDI service inside the package `GaudiSvc`. It was officially released in

2016 with GAUDI v27r1.

The service is developed as a C++ class which implements the following main methods:

- `isEnabled` captures information whether the service is enabled in the job or not.
- `start` initiates the service and calls `collectData`.
- `collectData` executes the main functionality of the service. It traverses and queries the GAUDI tools (`ToolSvc`), services (`Services`), algorithms (`Algorithms`) and Job options (`JobOptionsSvc`) to capture their configuration. This is demonstrated in Example 6.4.1.
- `getMetaData` returns the object that stores a dictionary of the job configurations.

The metadata object, named *info*, is implemented as a dictionary (a map in C++ i.e. `std::map`), where the keys capture the names of application configurations, and the values capture their information.

At the beginning of every job all internal job settings are empty. Therefore, the metadata can only be captured once the components and configurations are assigned to the job. For this purpose, we considered three application audit methods, which follow the job execution. They are automatically invoked by the Application manager at the start of every job. Those methods are:

- `initialize` that initialises algorithms and standard services, and sets job properties,
- `execute` that executes the main action of the job,
- `finalize` that is called at the end of the job.

The Metadata service could not have been called from either `initialize` or `execute` methods as not all job information are set during their execution. The metadata service is thus called to be executed from the `finalize` method, at the moment when output data has been written to a ROOT file. At this point, we are certain to capture all of the configurations and variables that were used in the job. The service is then completed when the metadata dictionary is saved in the ROOT file.

```
ApplicationMgr.AlgTypeAliases: {  },
ApplicationMgr.AppName: DaVinci,
ApplicationMgr.AppVersion: v50r0,
ApplicationMgr.HistogramPersistency: ROOT,
...
IAlgManager.Algorithms: Tuple, Tuple2, Tuple3,
ISvcLocator.Services: MessageSvc, JobOptionsSvc,
    RndmGenSvc.Engine, RndmGenSvc, Gaudi::MetaDataSvc,
    AppMgrRunable, IncidentSvc, EventPersistencySvc,
    EventDataSvc, AlgContextSvc, TimelineSvc, RootHistSvc,
    HistogramPersistencySvc, HistogramDataSvc, NTupleSvc,
    AlgExecStateSvc, EventLoopMgr, ToolSvc,
...
JobOptionsSvc.OutputLevel: 3,
JobOptionsSvc.PATH: ../options/job.opts,
...
MessageSvc.AuditFinalize: False,
...
NTupleSvc.AuditFinalize: False,
...
NTupleSvc.Output: [ "DATAFILE=TupleEx.root ... "],
...
ToolSvc: ''}
```

(a) A list of configurations collected, which demonstrates a wide range of service configurations covered in the Metadata service.

```
atrisovic — atrisovi@lxplus027:/afs/cern.ch/work/a/atrisovi/lhcb-metadatasvc/make-ntuple-example1 — ssh -YX atrisov...
[atrisovi@lxplus027:  dumpMetaData TupleEx.root
{'ApplicationMgr.ActivateHistory': 'False',
 'ApplicationMgr.AlgTypeAliases': '{  }',
 'ApplicationMgr.AppName': 'DaVinci',
 'ApplicationMgr.AppVersion': 'v50r0',
 'ApplicationMgr.AuditAlgorithms': 'False',
 'ApplicationMgr.AuditServices': 'False',
 'ApplicationMgr.AuditTools': 'False',
 'ApplicationMgr.CreateSvc': '[ ]',
 'ApplicationMgr.Dlls': '[ ]',
 'ApplicationMgr.Environment': '{  }',
 'ApplicationMgr.EventLoop': 'EventLoopMgr',
 'ApplicationMgr.EvtMax': '100',
 'ApplicationMgr.EvtSel': 'NONE',
 'ApplicationMgr.Exit': '0',
 'ApplicationMgr.ExtSvc': "[ 'RndmGenSvc' , 'Gaudi::MetaDataSvc' ]",
 'ApplicationMgr.ExtSvcCreates': 'True',
 'ApplicationMgr.Go': '0',
 'ApplicationMgr.HistogramPersistency': 'ROOT',
 'ApplicationMgr.InitializationLoopCheck': 'True',
 'ApplicationMgr.JobOptionsPath': '',
 'ApplicationMgr.JobOptionsPostAction': '',
 'ApplicationMgr.JobOptionsPreAction': '',
 'ApplicationMgr.JobOptionsSvcType': 'JobOptionsSvc',
```

(b) Retrieving metadata from command line. Only the first part of the metadata list is shown here.

Figure 6.2: Demonstration of Example 6.4.1

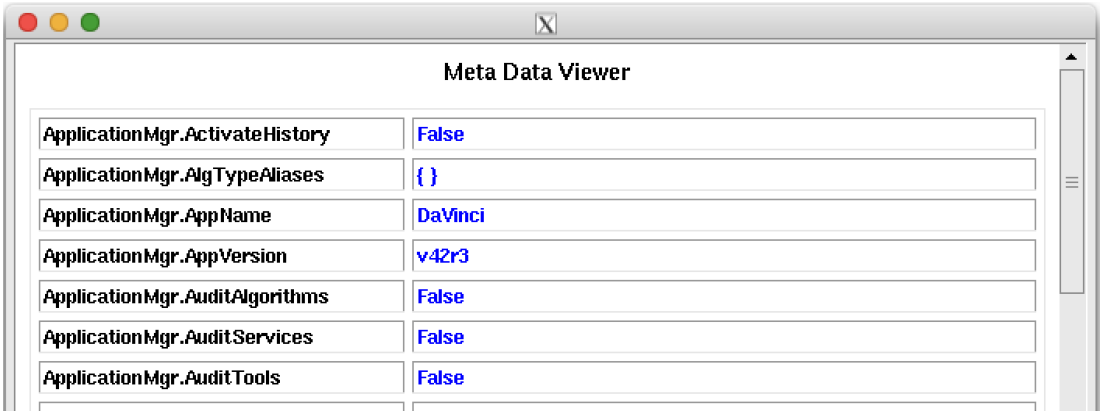
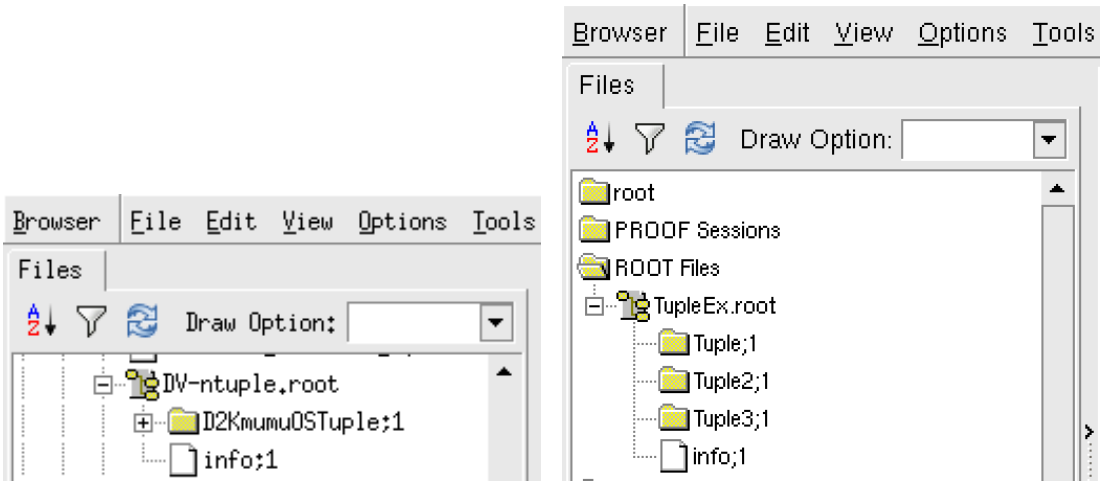


Figure 6.3: The HTML-based graphical user interface of the Metadata viewer.



(a) The ROOT ntuple produced from Example 6.5.1.

(b) Ntuple shown in Example 6.4.1.

Figure 6.4: After an ntuple has been processed using the provenance tracking service, it stores an object with the metadata called info.

6.4.1 Provenance viewer

Clean and straightforward presentation of data provenance is an integral part of this contribution. The metadata dictionary can be seen via a command-line tool as `dumpMetaData <filename>` (as shown in Figure 6.2b), but it can also be seen from a stand-alone provenance viewer shown in Figure 6.3. It is implemented as a pop-up window based on C++ and ROOT. This means that it requires the ROOT framework (and an input ROOT ntuple file) to be executed. The viewer uses an HTML table to present the key-value pairs of metadata captured in the job.[3]

Example 6.4.1. The functionality of the Metadata service is demonstrated using an example

[3]Hypertext Markup Language (HTML) is the standard markup language for creating web pages and web applications [156].

available at Ref. [5]. In Figure 6.2a and Figure 6.2b, we can partially see the list of configurations that were collected by the Metadata service after the job execution. The appearance in the ROOT framework is shown in Figure 6.4b. The beginning of the list captures the Application manager, and then the algorithms, services, job options and default services such as the message service, the ntuple service and the tools service. We can also see that the job was configured to produce an output file called `TupleEx.root` and that it used the application DAVINCI V50R0.

6.5 Usage of the service

In order to use an LHCb application, analysts need to provide application configurations that are normally passed in a python file. For the DAVINCI application, these application configuration files can also include selecting particle decays and applying filters on the data. The Metadata service is not used by default in the GAUDI framework, but it can be enabled in the application configuration file. This is done by assigning an external service (`ExtSvc`) to the Application manager, with the following command:

```
ApplicationMgr().ExtSvc += [ 'Gaudi::MetaDataSvc' ]
```

Example 6.5.1. Consider an ntuple describing the decay of $D^+ \to K^+\mu+\mu-$ that was produced with DAVINCI V42R3. The ntuple is captured with an additional *info* object produced by the Metadata service, as shown in Figure 6.4a. Its configuration is shown in Figure 6.3.

The GAUDI framework has a particular feature which enables job execution using a "flat" list of options, instead of the application configuration file. This list of options is equivalent to the configurations in the python application configuration file, but the format of the file is different as the options are given line by line in the file. The list can be stored in a python file, python pickle file or Linux configuration options file. All of these file formats are used for serialising and deserialising python objects to be saved or read from the disk.

If there is a need to reproduce a ROOT ntuple, the original job can be recreated by extracting and executing metadata dictionary captured in the ntuple as a "flat" list of options in GAUDI. Finally, even though DAVINCI is most commonly used for the ROOT ntuple production, the Metadata service can be used for other LHCb applications that can produce ROOT ntuples like for example BRUNEL (as shown in Figure 6.5).

```
AlgContextSvc.BypassIncidents= True,
All.MeasureTime= True,
All.Members= ['Gaudi::Hive::FetchDataFromFile/FetchDSTData',
'createODIN/ODINFutureDecode', ...],
ApplicationMgr.ActivateHistory= False,
ApplicationMgr.AlgTypeAliases= {  },
ApplicationMgr.AppName= 'Brunel',
ApplicationMgr.AppVersion= 'v60r0',
ApplicationMgr.AuditAlgorithms= True,
ApplicationMgr.AuditServices= False,
ApplicationMgr.AuditTools= False,
ApplicationMgr.CreateSvc= [  ],
ApplicationMgr.Dlls= [  ],
ApplicationMgr.Environment= {  },
ApplicationMgr.EventLoop= 'EventLoopMgr',
ApplicationMgr.EvtMax= '100',
ApplicationMgr.EvtSel= '',
ApplicationMgr.Exit= '0',
ApplicationMgr.ExtSvc= [ 'DetDataSvc/DetectorDataSvc' ,
'ToolSvc' , 'AuditorSvc' , 'Gaudi::MetaDataSvc' ,
'Gaudi::MultiFileCatalog/FileCatalog' , ...],
...
```

Figure 6.5: A "flat" list of options collected by the Metadata service, which can be used to reproduce an ntuple. The figure does not show all captured options.

6.6 Chapter summary

In this chapter, I explained the process of creating the ROOT ntuples. The analysts write application configuration files to retrieve datasets for individual analysis. At this point, the process of data production has passed, and this marks the beginning of data analysis.

Data provenance is one of the essential concepts in data preservation and reproducibility, as we have seen through several examples explored in this chapter. I helped improve provenance tracking in the LHCb software by developing a GAUDI service that captures application configurations when creating a ROOT ntuple. The captured job configurations can be used to reproduce the ROOT ntuple. This chapter presented the basic concepts of the GAUDI framework and the implementation of the Metadata service. The service was first released in GAUDI v27r1, meaning that it is available for use in physics analyses with DAVINCI v40r0 onwards.

In the following chapter, I retrieve a dataset from the Grid that describes the rare decays of $D^+_{(s)} \to \pi^+\mu^+\mu^-$. Following that, I attempt to reproduce once published results that studied these decays.

Chapter 7

Reproducing an LHCb physics analysis

Reproducing results is an integral part of performing physics analyses. Physics measurements obtained from one dataset are often cross-checked using another dataset. For example, if an analysis is performed on 2011 and 2012 data (Run 1), its results might be verified using only the 2011 data. However, when a third party tries to reproduce HEP analyses, the situation is not that simple. The challenges lie in large datasets, complex software and hardware dependencies, but also there are issues that cannot be anticipated in advance. It is necessary to attempt to reproduce a physics analysis as a third party, in order to reveal the obstacles and barriers. Finding a way to overcome these barriers is the only way to ensure analysis reproducibility.

In this chapter, I describe an attempt to reproduce a physics analysis at LHCb. It is important to note that while doing this, we are not in contact with the proponents of the original analysis, and we do not use their code or ntuples. To guide the study, we first considered using solely publicly available physics papers. However, it was immediately clear that the papers focus on physics results and that they lack technical information needed to reproduce them. For this reason, and to achieve a higher level of reproduction, we turn to all available documentation both public and internally available. The goal of this study is to evaluate how successfully we can reproduce an analysis and to identify potential barriers. In the chapter that follows, I discuss ways in which we can ensure that our analysis code and methods are fully preserved and reproducible.

7.1 Search for rare decays of the charm meson

With the goal of selecting a standard LHCb analysis, finding the available documentation and then attempting to reproduce it, we chose an analysis performed on the rare decay $D^+_{(s)} \to \pi^+\mu^+\mu^-$ and its related modes [157]. The choice of this particular analysis was made partly due to the expertise on $\mu\mu$ decays present in the HEP group in Cambridge. However, the exact choice of $D^+_{(s)} \to \pi^+\mu^+\mu^-$ is unimportant. From the available documentation, we have a thesis (Ref. [158]), a paper (Ref. [157]) and an analysis note (Ref. [159]). This documentation presents searches for the same decay within two different datasets recorded at LHCb. We consider the thesis our primary source of information, and we attempt to reproduce those results. Only when descriptions of analysis methods are not clear in the thesis, we reach for the analysis note or other available information. Analysis notes are internal collaboration documents that are linked to the published paper, but they often offer more technical details on analysis implementation and data provenance.

This study aims to identify both good and bad practices in performing analyses. Therefore it is crucial that the reader understands that we do **not** want to criticise any scientific work at LHCb, but merely suggest improvements to the common practice.

The theoretical background and a complete phenomenological motivation for the decay mode $D^+_{(s)} \to \pi^+\mu^+\mu^-$ is laid out in the public documentation at Ref. [157, 158] and it is not discussed here.

7.2 Reproducing data selection

The original analysis took place in 2012, with data collected in 2011, as presented in the paper (Ref. [157]) and analysis note (Ref. [159]). Following this, the analysis was repeated on the full Run 1 dataset and published in the PhD thesis at Ref. [158]. To reproduce the study, we use the full Run 1 dataset collected in *pp* collisions. This means that the data was collected in both 2011 and 2012 at the centre-of-mass energy of 7 TeV and 8 TeV respectively. The data was recorded during the stable detector conditions when all the components were fully operational, and the VELO was closed.

The decays of $D^+_{(s)} \to \pi^+\mu^+\mu^-$ are rare, which makes their observation extremely challenging. The goal of this study is to observe the decay or set limits on the branching fraction. The previously found limits on the branching fractions are: $\mathscr{B}(D^+ \to \pi^+\mu^+\mu^-) <$

3.9×10^{-6} [160] and $\mathscr{B}(D_s^+ \to \pi^+\mu^+\mu^-) < 2.6 \times 10^{-5}$ [161].

It is important to mention that there are other $D_{(s)}^+$ 3-body decays that reach the same decay final state of stable particles as $D_{(s)}^+ \to \pi^+\mu^+\mu^-$. The $D_{(s)}^+$ can decay into three particles π^+, μ^+ and μ^- directly, or it can decay via resonances into $\eta\pi^+$, $\rho\pi^+$, $\omega\pi^+$ or $\phi\pi^+$. Later η, ρ, ω, ϕ would decay into $\mu^+\mu^-$ reaching the same final state. The resonances with their branching ratios are shown in Table 7.1. The signal that we are looking for is solely $D_{(s)}^+ \to \pi^+\mu^+\mu^-$ without any $\pi^+\mu^+\mu^-$ resonances. The branching fraction of $D_{(s)}^+ \to \pi^+\mu^+\mu^-$ is measured relative to a known and observed decay $D_{(s)}^+ \to \pi^+(\phi \to \mu^+\mu^-)$, which is called a normalisation channel (or control mode).

Final state	Resonances	Branching ratio
$\pi^\pm\mu^+\mu^-$	-	-
	$\eta\pi^+$	$(9.0 \pm 1.7) \times 10^{-8}$
	$\rho\pi^+$	$(9.1 \pm 5.5) \times 10^{-9}$
	$\omega\pi^+$	$(2.07 \pm 0.89) \times 10^{-7}$
	$\phi\pi^+$	$(1.29 \pm 0.14) \times 10^{-5}$

Table 7.1: Branching fractions of resonances in $\pi^\pm\mu^+\mu^-$ decay final state. [159].

The LHC produces a lot of charm particles, but it also produces other particles, some of which can be mistaken for the signal. It is necessary to develop an effective strategy to reduce these backgrounds and identify the signal events in the large data sample. The event selection strategy is implemented in three stages: the trigger selection, stripping selection and the multivariate (final) selection.

7.2.1 Trigger Selection

The trigger selection (introduced in Chapter 2.4) is applied during data-taking. Both our datasets and the datasets used in the original analysis were collected in 2011 and 2012 at LHCb when the same trigger system was running. However, the trigger selection depends on the stripping lines (introduced in Chapter 4) which can place requirements on trigger lines. Even if we use a different stripping version than the original analysis, it is probable that the trigger selection would be the same between the strippings, however, we cannot be certain. This is further discussed in the following section.

Roughly speaking, the Run 1 trigger that we use in this study would select muons in the L0 trigger, and the high-level trigger would identify and select charm hadrons. The cuts of the

trigger selection are laid out in the public documentation available at Ref. [158].[1]

7.2.2 Stripping Selection

Choosing the right stripping line was the first challenge while performing this study. The original analysis published in the paper was performed using data collected in 2011 from STRIPPING17 [159]. However, there are no indications what stripping version was used in the thesis, except for a table showing stripping cuts. More than ten new stripping versions have been released since STRIPPING17, and in order to find the right dataset, the selection cuts need to be compared. The cuts presented in the table resemble the ones applied in STRIPPING17. However, the original proponents could not have used only STRIPPING17 selection as it contains data collected in 2011 but not 2012 [162]. It remains unclear what stripping version was used in the thesis, as it was impossible to deduce from the documentation.

We choose to use the current recommended stripping, which is STRIPPING21 and STRIPPING21R1. To illustrate the possible differences between the strippings, the comparison of the stripping cuts of STRIPPING17 and STRIPPING21 is shown in Table 7.2. However, we assume that the potential small differences in stripping selection in between the datasets should not affect the result, as the expectation is that they will be minimised with the following tighter selection on the data in the analysis (explained in the following sections).

The input data is retrieved from the same stripping line as used in the original study, which is `D2XMuMu_PiOS`. We use both the data taken with the magnetic field polarity directed towards the positive y-axis (magnet up) and the data taken with magnet polarity directed towards the negative y-axis (magnet down). These datasets are merged at the start of the analysis and not analysed independently.

7.2.3 Monte Carlo samples

The Monte Carlo samples are used to model the signal in this study. They were generated using both 2011 and 2012 beam settings, which correspond to STRIPPING21R1 and STRIPPING21. The datasets were split into approximately 500k magnet down and 500k magnet up events for 2011, and 1M of magnet down and 1M of magnet up events for 2012, as shown in Table 7.3. The events were generated using the PYTHIA 8 [132] generator package, and the detector response was modelled using the GEANT4 [134, 135] toolkit.

[1] The threshold (or filter) values are commonly called *cuts*, and they are applied to a dataset to extract desirable candidates.

Cut		STRIPPING21	STRIPPING17
$\mu^{\pm} Track \chi^2_{NDOF}$	$<$	5	8
$\mu^{\pm} p$	$>$	3000.0 MeV	2000.0 MeV
$\mu^{\pm} p_T$	$>$	500.0 MeV	300.0 MeV
$\mu^{\pm}$ Minimum impact parameter χ^2	$>$	6	6
$\pi^{+} Track \chi^2_{NDOF}$	$<$	5	8
$\pi^{+} p$	$>$	2000.0 MeV	3000.0 MeV
$\pi^{+} p_T$	$>$	300.0 MeV	500.0 MeV
π^{+} Minimum impact parameter χ^2	$>$	6	4
D^{+} decay vertex χ^2_{NDOF}	$<$	5	5
D^{+} impact parameter χ^2	$<$	25	30
D^{+} DIRA^2	$>$	0.9999	0.9999
$\pi\mu\mu$ invariant mass from PDG value	$<$	200.0 MeV	200.0 MeV
D^{+} maximum distance of closest approach	$<$	0.15	
$\pi\mu\mu$ invariant mass	$>$	1763.0 MeV	1763.0 MeV
$\mu\mu$ invariant mass	$>$	250.0 MeV	250.0 MeV

Table 7.2: The table shows the difference in data selection between STRIPPING21 and STRIPPING17.

Production ID	Year	Magnet polarity	Processing pass	Number of events
12702	2011	Down	Reco14a/Stripping20r1	0.5M
12701	2011	Up	Reco14a/Stripping20r1	0.5M
11621	2012	Down	Reco14/Stripping20	1M
11624	2012	Up	Reco14/Stripping20	1M

Table 7.3: Monte Carlo simulation samples used in the analysis. The production ID (from DIRAC) uniquely identifies the production process of the samples.

The thesis does not provide much information about the choice of Monte Carlo samples, however, we found a Monte Carlo production file in the DIRAC database where the names of the original proponents were featuring, which was a good indication that this sample was used in the original analysis. The DIRAC database is also an LHCb internal resource, but it is not very commonly used among the physics analysts. Therefore, if an LHCb analyst tried to reproduce this study, it is possible that they would not be able to find this clue.

Regarding the analysis note that should provide us with additional information, we note that the description of the Monte Carlo samples was not sufficient to identify them. It states that they use "MC10" and that this sample "produce[d] superior results to MC11" sample [159]. However, we are unable to identify the dataset when searching through the LHCb bookkeeping system.

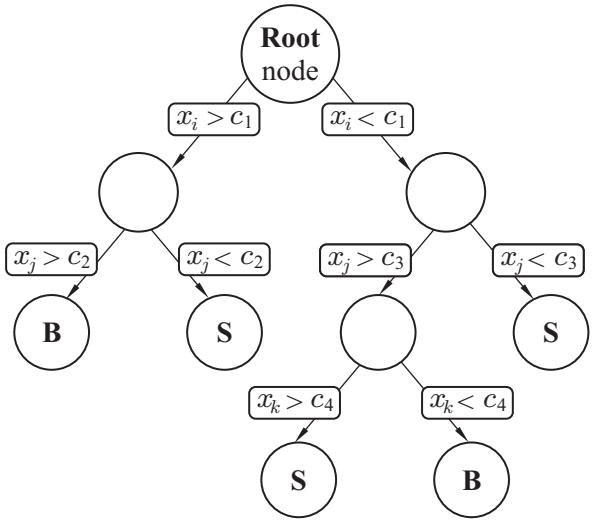


Figure 7.1: Schematic view of a decision tree adopted from Ref. [163]. The variables x_i of the dataset are filtered through a series of binary splits starting from the Root node. At each node, a feature cut (here denoted $c_{1,2,3,4}$) is used, which provides the best separation between signal and background. The same features may be used multiple times at several nodes, while others might not be used at all. The leaves of the tree are labelled "S" for signal and "B" for background depending on the majority of events that end up in the respective nodes.

7.3 Reproducing the multivariate selection

Multivariate selections, based on machine learning approaches, are commonly used in physics analyses. They use classification algorithms to make predictions in data and distinguish signal events from the background in real data samples. When the algorithm is constructed using a training sample whose category is already known, this is then called supervised machine learning. Typically, Monte Carlo samples are used for the training, though depending on use case, sometimes the signal also can be defined by real data. The algorithm is then used to classify unseen inputs based on their similarity to the training sample.

7.3.1 Boosted decision trees

Decision trees (DT) are one of the machine learning algorithms that has proven to be successful in the selection of event candidates in HEP. Its structure is a binary tree as shown in Figure 7.1, which is created during the training on a pure sample of signal and background events. During the classification, for each new input, a series of binary decisions are made by sorting the data at each step. The selection starts at the root node and events that pass the condition are sorted in one way, and the ones that do not the other. This is performed until the event is classified by reaching a stop criterion at a leaf of the tree.

"Overfitting" or "overtraining" is a problem which arises when the selection algorithm captures random noise instead of the underlying relationships in the data. Such selection has poor predictive performance, as it overreacts to minor fluctuations in the training data to produce suboptimal results. The instability of a decision tree is overcome by creating a large number of decision trees forming a *forest*. The event is then classified by considering the response of each tree and returning the majority vote. All trees in the forest are trained on the same training sample. The events misclassified by the first tree are given a higher weight in training the second tree and so forth. This process is called boosting, and it increases the statistical stability of the classifier remarkably improving the separation performance compared to a single decision tree [163]. After the trees have been trained, they are combined together with these weights to create a single classifier called a Boosted Decision Tree (BDT).

In this study, we use the Toolkit for Multivariate Analysis (TMVA) [163], which is a machine learning software package integrated into the ROOT framework. TMVA is often used for machine learning in HEP, and it is commonly used by all the LHC experiments. The pure signal was obtained from Monte Carlo samples and it was defined around the D^+ invariant mass (at the cut on invariant mass values of `D_MM>1840` and `D_MM<1900`), as described in Ref. [158]. The background was supplied from the real data outside of the signal region (at the cut on invariant mass values of `D_MM>2010`) as shown in Figure 7.2 in red.

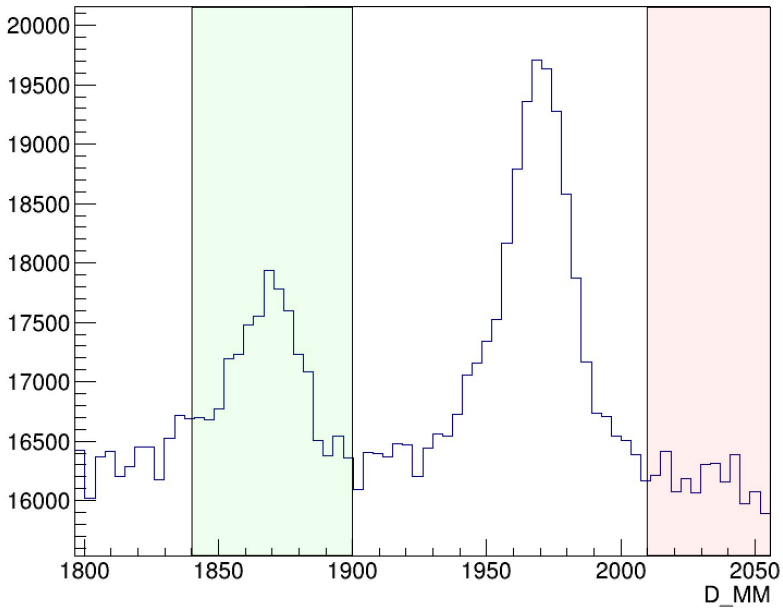


Figure 7.2: Invariant mass of the D^+ and D_s^+ mesons in real data sample. For the BDT training we define signal from the Monte Carlo sample in the region marked in green in the figure (only D^+), and background from real data in the region marked in red.

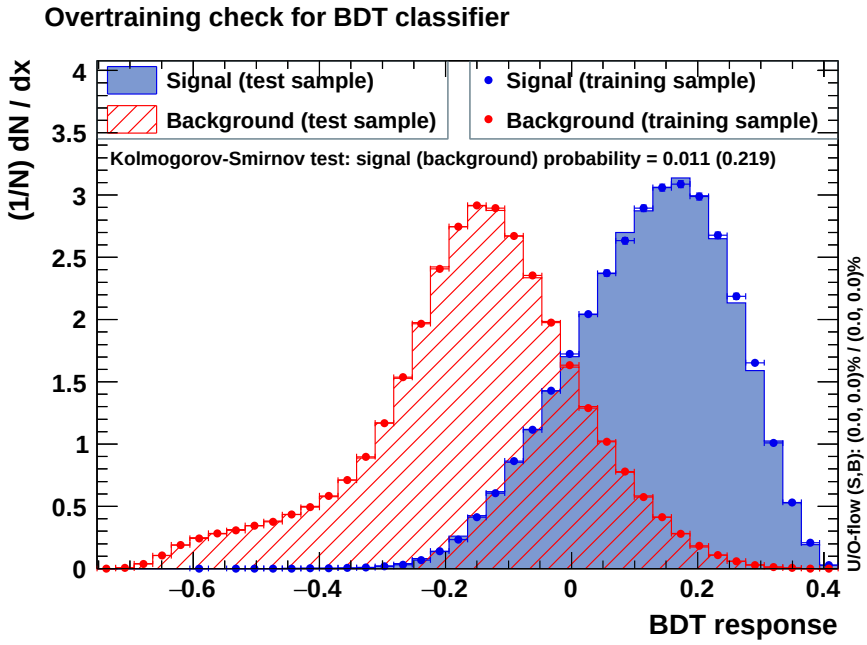
Using TMVA, a range of classification techniques were compared. The BDT classifier with the adaptive boost algorithm was found to provide the best background rejection and signal efficiency. In the original study, TMVA was also used for multivariate selection, as explained in Ref. [159] and Ref. [158], but only cited in Ref. [157]. However, in the original study, the gradient boost algorithm was found to produce optimal results. The TMVA response for the BDT is displayed in Figure 7.3a. The ROC curve that shows the background rejection versus signal efficiency for both adaptive and gradient boost algorithms is shown in Figure 7.3b.

The candidates are selected based on available kinematic and particle information. The BDT uses the following variables to discriminate the background. Because the mapping between variables in the data and the thesis was not intuitive, this list is not completely identical to the one presented in the documentation.

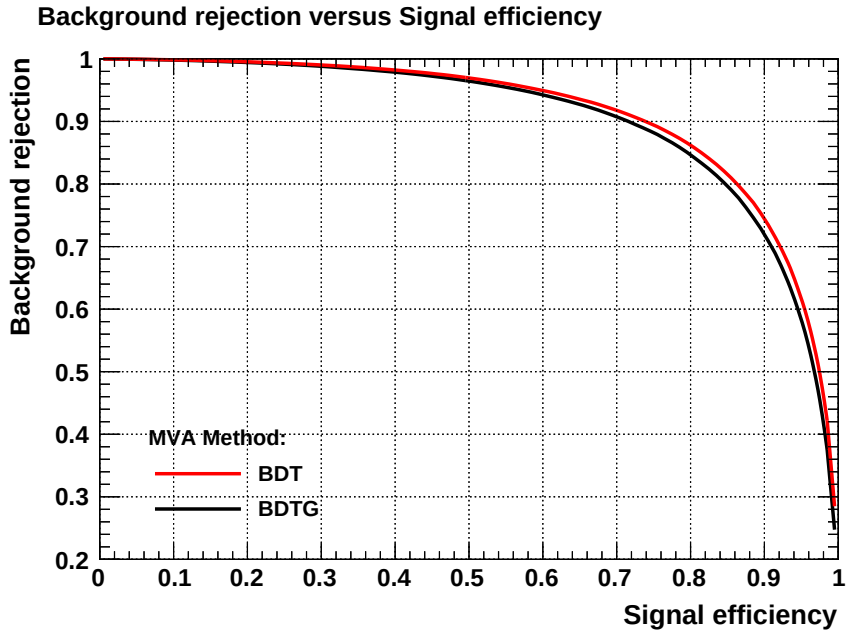
- The transverse momentum p_T of D^+ ($p_T(D^+)$), muons ($p_T(\mu^\pm)$) and pions ($p_T(\pi^+)$). p_T is defined as the component of a particle's momentum perpendicular to the beam direction.
- The impact parameter of D^+ ($\chi^2_{IP}(D^+)$), muons ($\chi^2_{IP}(\mu^\pm)$) and pions ($\chi^2_{IP}(\pi^+)$). IP is the distance between the pp interaction point and the nearest point on the particle trajectory.
- The proper lifetime of D^+ denoted as $\tau(D^+)$. It is defined as a time period from when the particle is created to when it decays, in the D^+ rest frame.
- The $cos(\theta_D)$ angle between the D^+ candidate momentum, and the line connecting the secondary and primary vertex (DIRA).
- The end-vertex (EV) ($\chi^2_{EV}(D^+)$).
- The flight distance (FD) of D^+ ($\chi^2_{FD}(D^+)$).
- The particle momentum p of D^+ ($p(D^+)$), muons ($p(\mu^\pm)$) and pions ($p(\pi^+)$).

TMVA is a complex software that offers a wide range of possibilities for machine learning training and has hundreds of machine learning features. In analysis preservation, this information must be documented. However, there are three ways in which a TMVA selection can be bit-wise reproduced:

- using the same code,
- using the same input parameters or



(a)



(b)

Figure 7.3: (a) TMVA response for BDT. (b) ROC curve. The BDT line employs adaptive boost algorithm and it has better performance than BDTG line, which uses gradient boost algorithm and was found to be the best method in the original study.

- using the same training output (XML files).

We have not had access to any of these resources. Furthermore, it is necessary to use the exact same version of the ROOT framework, as newer versions often have new features or they solve some previous issues in the code that may affect the output. However, the information about ROOT version was also not documented. Due to this complexity, we already observe that the optimal classification algorithms are different, and we expect discrepancies between the original and our results.

7.3.2 Selection optimisation

The selection optimisation is a process where we compare combinations of different variable cuts applied to the data in order to maximise the performance of the analysis. In this analysis, and likewise in the original analysis, the optimisation is performed considering the values of PID of the two muons and the BDT. PID is a value produced from a process of labelling the particle according to its kinematic features, as described in Section 2.3.2. The figure of merit to describe analysis performance is called significance, and we use the same formula to define it as it was used in the original study, available in Ref. [159]. The formula is:

$$Significance = \frac{s}{\sqrt{s+b}}, \tag{7.1}$$

where s and b stand for the selected signal and background respectively. The signal yield s is computed using the theoretical branching ratio $\mathscr{B}(D \to \pi\mu\mu) = 3.7 \times 10^{-9}$ [164] corrected by the normalisation factor derived using the $D \to \phi\pi$ channel.

$$s = \frac{\mathscr{N}(D \to (\phi \to \mu\mu)\,\pi) \times \mathscr{B}(D \to \pi\mu\mu)}{\mathscr{B}(D \to \phi\pi) \times \mathscr{B}(\phi \to \mu\mu)} \tag{7.2}$$

The background yield b is supplied from the normalisation channel ϕ as the extrapolated background in the signal region of $\mu\mu$ invariant masses.[3] In the original study, b was supplied by "multiplying the extrapolated background in the signal region by a phase-space coefficient". The explanation of the coefficient was ambiguous, hence we needed to make an educated assumption to proceed with the study. For each combination of $\mu\mu$ PID (PIDmu) and BDT cuts, the significance is calculated and presented in the heat map in Figure 7.4. The optimal cuts are found to be at BDT >0.1 and PIDmu >2. In the original analysis, the optimal cuts were found

[3]The invariant mass is the portion of the total mass of an object or system of objects that is independent of the overall motion of the system[165].

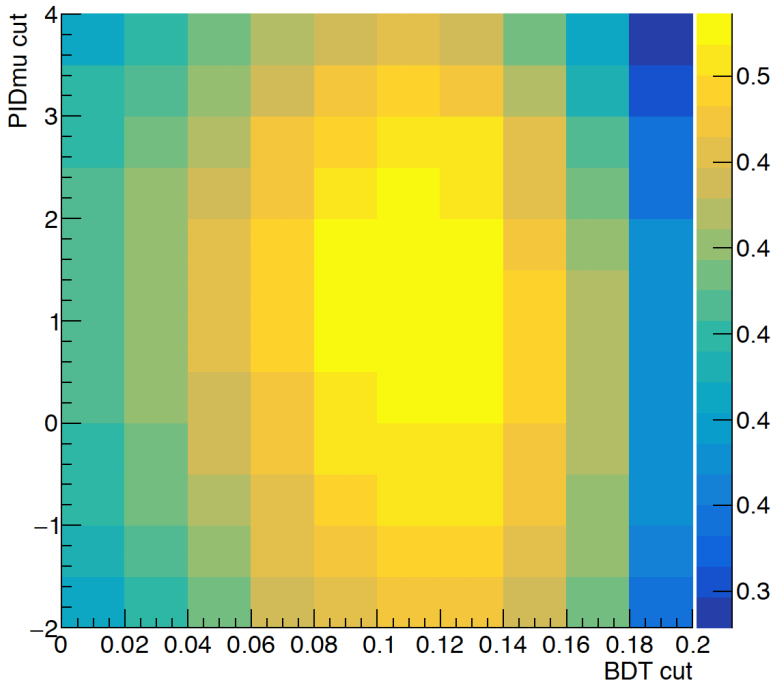


Figure 7.4: The optimisation plot, which shows significance for each combination of the BDT and PIDmu cuts.

to be at BDT >0.98 and PIDmu >3, which indicates that the selection was very different. These differences and their causes are discussed in the following sections. The final selection of cuts is shown in Table 7.4. In addition to the cuts obtained with optimisation, we applied cuts on the probability of misidentification of π^+ as K or μ as described in the thesis [158].

7.4 Reproducing invariant mass fit results

Using our optimised cuts, the data is split into bins of dimuon invariant mass ($m(\mu^+\mu^-)$, also denoted as q^2), to identify the resonances that reach the final state of $D^+_{(s)} \to \pi^+\mu^+\mu^-$. The signal shapes are fixed from the simulation sample of the ϕ invariant mass. The boundaries of each bin are shown in Table 7.5 and the invariant mass of dimuon with resonance boundaries is shown in Figure 7.5.

The D^+ and D_s^+ signal was modelled with the sum of Crystal Ball distributions ("double Crystal Ball"). Each shape consists of a Gaussian core with a power law tail on opposite sides. The background was modelled with a 2nd order Chebyshev polynomial distribution, as described in documentation [158]. The first peak featuring in the figures at the mass of $1870\,\mathrm{MeV}/c^2$ is the D^+ signal, while the second peak at the mass of approximately $1975\,\mathrm{MeV}/c^2$ is the D_s^+ signal. The unbinned extended maximum likelihood fit is performed, and the resulting mass fit

Particle/Event	Requirement
Event	BDT>0.1
π^+	`piplus_PIDK` <0 `piplus_PIDmu`<0
$\mu^\pm$	`muplus_PIDmu`>2 `muminus_PIDmu`>2

Table 7.4: Criteria of our final selection.

Bin description	$m(\mu^+\mu^-)$ range [MeV/c^2]	Figure
low-m($\mu^+\mu^-$)	250 - 525	7.8d
η	525 - 565	7.8b
ρ^2/ω	565 - 850	7.8e
ϕ	850 - 1250	7.8a
high-m($\mu^+\mu^-$)	1250 - 2000	7.8c

Table 7.5: Dimuon resonances and their invariant mass range.

plots are shown in Figure 7.7, and the respective plots of the original study are shown in Figure 7.6 (taken from Ref. [158]). Our results, however, look very different than the original plots as they have around three times more signal and ten times more background. This difference is discussed in the following section.

There are two peaking backgrounds which affect the analysis. The first is residual misidentified $D^+_{(s)} \to \pi^+\pi^+\pi^-$, where pions were misidentified as muons, and the second is partially reconstructed $D^+_{(s)} \to \pi^+(\eta \to \mu^+\mu^-\gamma)$. The contribution from $D^+_{(s)} \to K^+\pi^+\pi^-$ decays are negligible because the branching fraction is small and it only partially falls within the fit region.

$D^+_{(s)} \to \pi^+\pi^+\pi^-$ have a significant branching fraction. In the invariant mass fit to signal, PID cuts are placed on both muon candidates to suppress this background. However, the peaking background from misidentified $D^+_{(s)} \to \pi^+\pi^+\pi^-$ is visible just below the D^+_s mass in the high/low dimuon signal plot shown in Figure 7.6d and 7.6c.

There is no significant evidence for observation of the signal in the studied range of high and low dimuon mass, as per the original analysis. At this point the results have vastly diverged and, considering the available documentation, it was clear that they would not improve in the following steps. Therefore, systematic uncertainties were not studied as they would not bear much resemblance to the original.

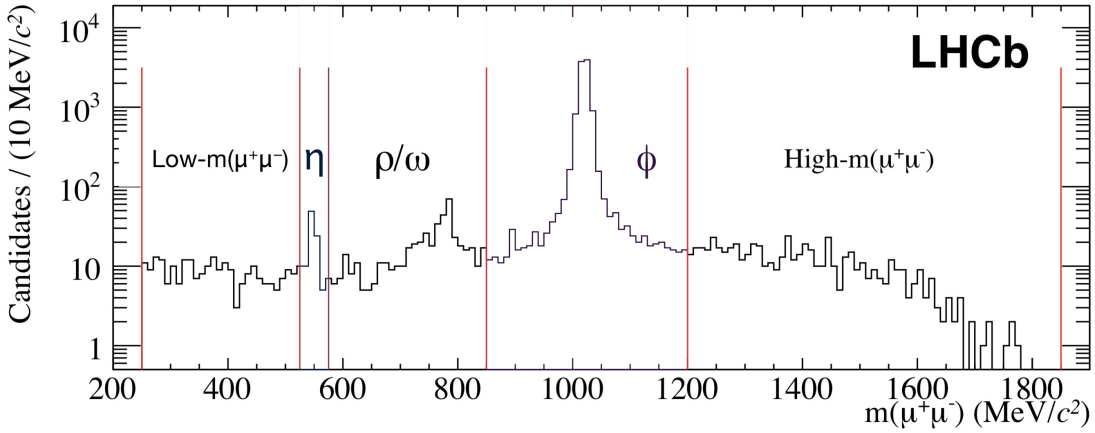


Figure 7.5: Invariant mass of the two muons with resonance boundaries. Normalisation channel: $D^+_{(s)} \to \pi^+(\mu^+\mu^-)_\phi$ is shown in the center of the figure. Taken from Ref. [166].

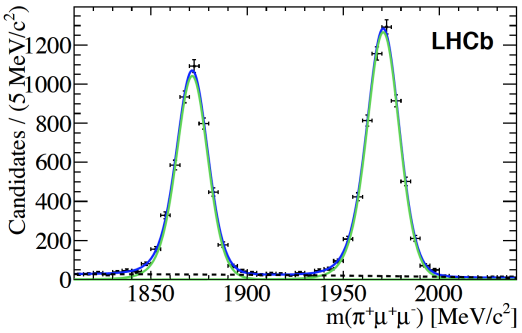
Particle/Event	**Requirement**
Event	BDT>0.17
π^+	`piplus_PIDK` <0 `piplus_PIDmu`<0
$\mu^\pm$	`muplus_PIDmu`>2 `muminus_PIDmu`>2

Table 7.6: Criteria of our manual selection.

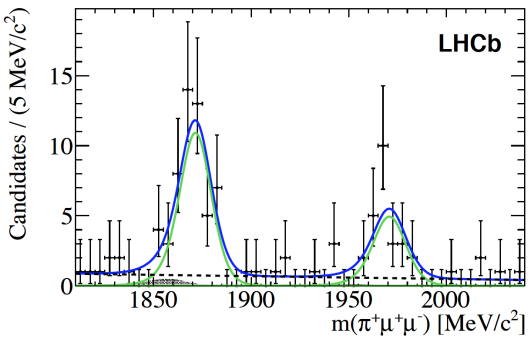
7.5 Discussion

As presented in the previous sections, it is clear that our final selection is much looser than the original final selection, which resulted in obtaining a very different result. Thus, in an attempt to understand this difference and improve our result, we apply a tighter selection considering the final cuts from the original study. First, we investigated different stripping cuts, in particular, the cuts from STRIPPING17 on pion p_T and p that are tighter than in STRIPPING21 (thus eliminating a higher number of candidates). After applying the cuts, we have not observed any reduction of background. Second, we try to manually tune the final selection to evaluate whether it could bring us towards the original plots. A tighter cut on BDT did indeed result in a better reproduction of the characteristics of the original plots. These updated plots are shown in Figure 7.8, and the new final selection is shown in Table 7.6.

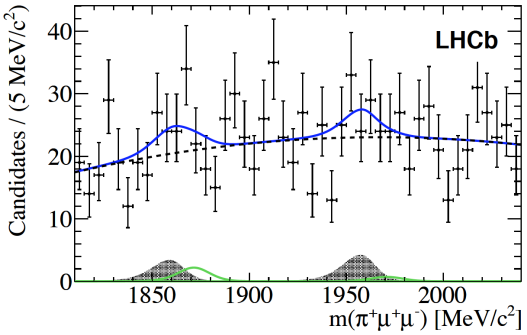
We conclude that reproducing an LHCb physics analysis is a significant challenge. Our aim was to recreate analysis results by following the available documentation. The original analysis was performed by many people over the course of several years, and the methodology is



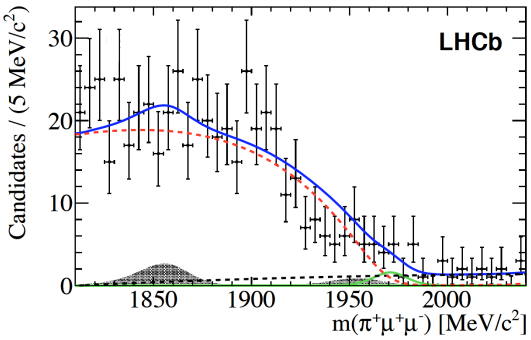
(a) $D^+_{(s)} \to \phi\pi^+$ resonance.



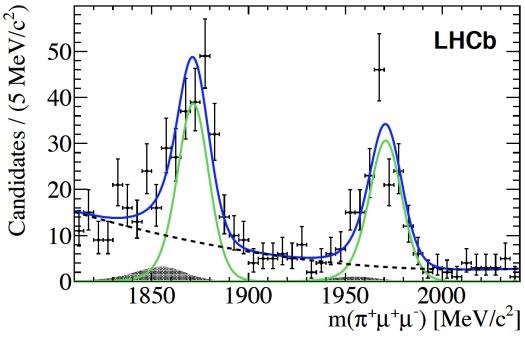
(b) $D^+_{(s)} \to \eta\pi^+$ resonance.



(c) High dimuon region.

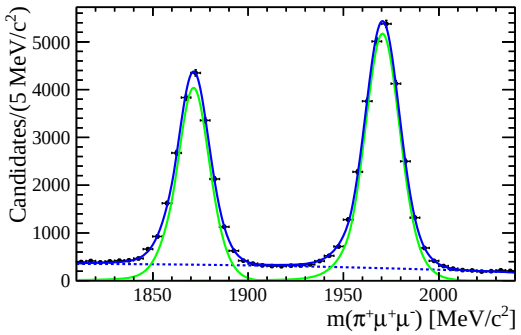


(d) Low dimuon region.

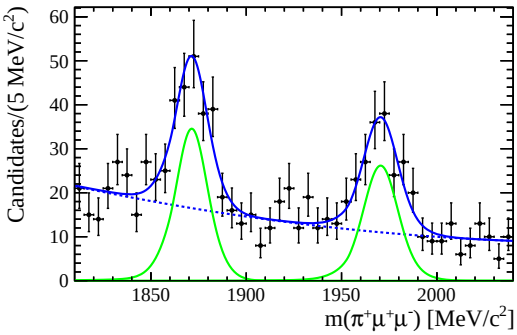


(e) $D^+_{(s)} \to \pi^+\rho/\omega$ resonance.

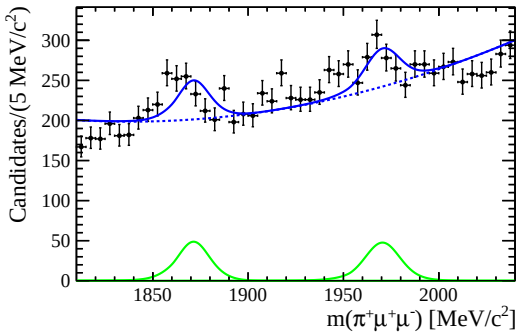
Figure 7.6: Plots from the original analysis [158]



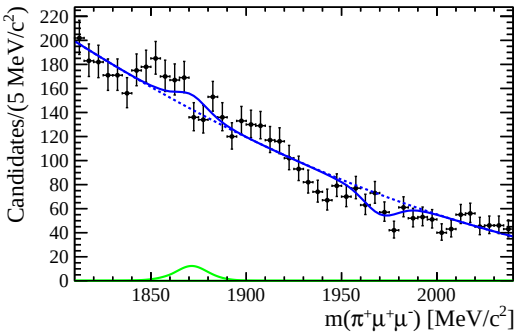
(a) $D^+_{(s)} \to \phi\pi^+$ resonance.



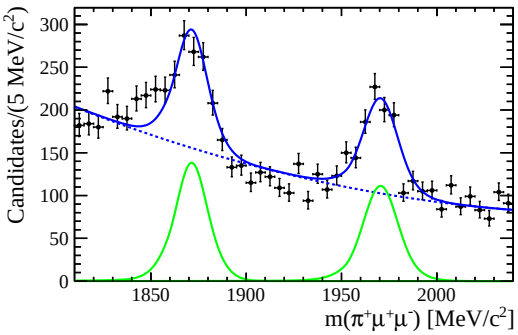
(b) $D^+_{(s)} \to \eta\pi^+$ resonance.



(c) High dimuon region.

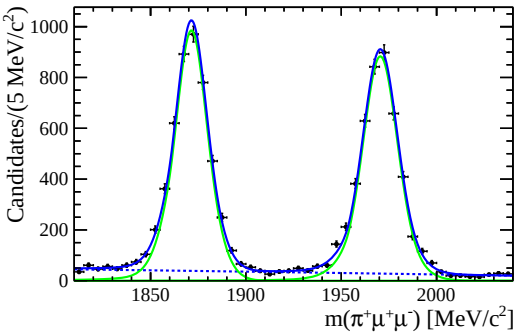


(d) Low dimuon region.

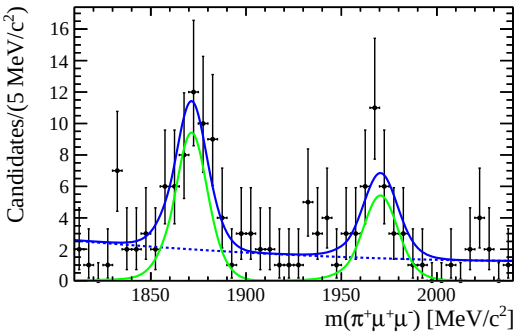


(e) $D^+_{(s)} \to \pi^+\rho/\omega$ resonance.

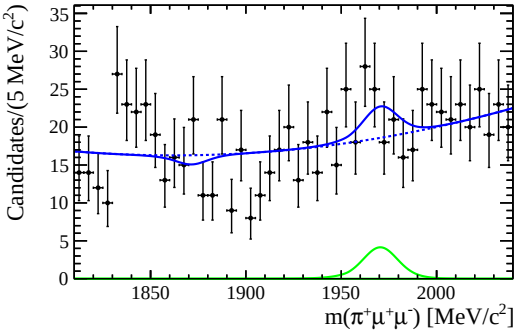
Figure 7.7: Plots obtained with the final selection.



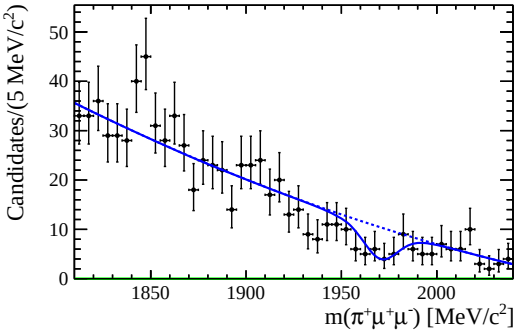
(a) $D^+_{(s)} \to \phi\pi^+$ resonance.



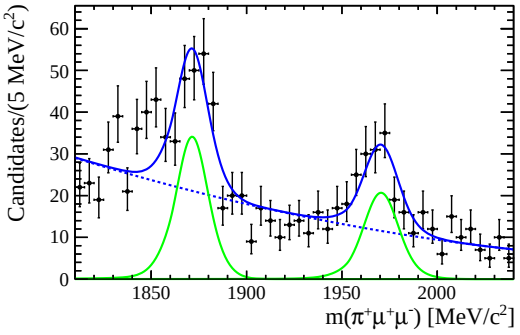
(b) $D^+_{(s)} \to \eta\pi^+$ resonance.



(c) High dimuon region.



(d) Low dimuon region.



(e) $D^+_{(s)} \to \pi^+\rho/\omega$ resonance.

Figure 7.8: Plots with manually found cuts.

complex and possibly dependent on contextual information, which is something very demanding to reproduce if it is not well-documented.

The discrepancies in the results are caused by a number of factors. The initial difference might have started with data selection. We could not identify what stripping version was used, therefore there may have been differences in the data at the beginning. We advocate clear documentation on what datasets were used, as thus far, this is not a formal requirement many analysts tend to exclude this information. The main difference was however introduced in the multivariate selection as the BDT configuration has many different options and parameters that could be employed in the training process, but here were not adequately documented. Furthermore, the version of the TMVA software that was used was also not documented.

Throughout the documentation, we note that there is a lot of specialised terminologies that were often obscure, and that mapping between variables in the data and descriptions in the documentation was not intuitive. This contributed to further divergence between our results and the original results. We encourage analysts, when writing the documentation, to consider that someone may attempt to reproduce their study. In some stages of this study, we were required to make educated assumptions, which again led to further discrepancies.

We also note that there were no indications of any preservation methods in the analysis documentation and no evidence that the analysis code was published to the collaboration. Therefore, all code used in this study to analyse the data was independently developed.

7.6 Chapter summary

To improve scientific documentation with an emphasis on reproducibility and preservation, we need to study the current practice. This chapter presented an attempt to reproduce a physics analysis using only available documentation at LHCb to learn about the methods used in the analysis. It is important to mention that we had access to the internal analysis note and databases such as DIRAC. This would not have been possible for a scholar outside of the LHCb collaboration, meaning that their only source of information would be the published paper and the thesis, which provides even less information than we had.

Reproducing a physics analysis has never been conducted before at LHCb in this manner, and it was a test for the previous analysis preservation efforts. We identified a number of barriers to analysis reproducibility introduced in the documentation, which can be solved with analysis preservation. The ambiguity found studying the documentation is not due to errors in

the original analysis, but it points to the faults of the current approach in documenting analysis. Notwithstanding, they are present not only in LHCb but any scientific field [31, 167–169]. Analysis preservation aims to enable analyses execution in the exact same way as the original by third parties who could or could not be physics experts.

The code developed to reproduce the analysis is published and documented at Ref. [6] to make these results reproducible.[4] In the next chapter, I introduce the best practices in performing data analysis and describe how resources of this analysis can be preserved and automated always to produce the same results.

[4]The code with input data can currently be executed by only the members of the LHCb collaboration, due to the strict LHCb data policies.

Chapter 8

Analysis preservation

Analysis preservation should be an integral part of performing analysis, as it facilitates better organisation of analysis resources, reproducibility and reuse. Nonetheless, there is no standardised way of capturing and preserving analyses, hence these practices are generally not applied.

In this chapter, I focus on the ways in which we can make data analysis fully preserved and reproducible. First, I introduce the pillars of analysis preservation, covering the preservation of the data, software and documentation. I explain how relevant and useful the tools used for preservation are in analysis development and collaborative work. Following this, the CERN Analysis Preservation framework is presented with its various features and tools. I use the analysis on $D^+_{(s)} \to \pi^+\mu^+\mu^-$ [6] performed in the previous chapter as an example to demonstrate analysis preservation and reuse.

8.1 Analysis preservation pillars

Analysis preservation attempts to cover the preservation of every analysis resource. This is why we define three analysis preservation pillars, which cover the preservation of input data, software and documentation. We also discuss the preservation of runtime environment and analysis workflows, as they are closely linked to the analysis software. The following sections address the current situation in the LHCb experiment regarding the pillars, and each of the sections ends with a discussion and presentation of a set of tools that can be used to facilitate resource preservation.

8.1.1 Preservation of the input data

There is no formal recommendation or requirement within LHCb on how to preserve the data once an analysis is completed. The tentative plan states that the data should be kept at least until the publication is released. This is because the paper reviewers could ask for changes to be made in the original work, which would make the reproduction of all datasets from the beginning impractical. However, after the publication, it is left to the analysts to keep or erase the data.

Due to the fact that there are strict collaboration policies applied to the LHC data access (as introduced in Chapter 1.6.2), the analysts cannot share their datasets in an open access repository, even though this is often encouraged in other scientific fields. The analysts should turn to the HEP-specific tools for sharing results and preserving the data.

HEPdata, as introduced in Chapter 1.7.3, is a free data repository used for presentation and preservation of the final results of HEP analyses. The service provides a form to submit datasets that are small in size, which are then visualised in an interactive way and published on the portal. For the larger datasets, the adequate solution for preservation is the CERN Analysis Preservation portal. The portal provides storage on the EOS system, which is already normally used in the LHCb physics working groups, meaning that the migration of the datasets should be effortless.

Generally speaking, there is a trade-off in data preservation in the fields dealing with large amounts of data. The benefit of keeping the initial datasets is that they can demonstrate the whole process of data filtering that leads to the final results. If the availability and cost of the storage allow for this solution, it is the one that we should choose. On the other side, if the resources are scarce, the preservation of the final datasets is sufficient. They can be used to reproduce the final result of an analysis, even though they cannot demonstrate the whole analysis process. The compromise is to be found in each analysis taking into account the available storage space.

Example 8.1.1. To put storage requirements into perspective, the initial dataset for the $D^+_{(s)} \to \pi^+\mu^+\mu^-$ analysis was approximately 54.1 GB of real data and 480 MB of Monte Carlo sample. These datasets were then filtered, and their size was reduced to a ROOT ntuple of 9.5 GB in total. In this case, the ntuple is not further reduced, and it is used to produce the final plots. Therefore, the storage of approximately 10 GB of space should be sufficient to capture the final steps of the analysis, while the storage of approximately 55 GB to preserve the whole workflow.

The CAP framework currently provides up to 1 TB per analysis for data storage, which may be a reasonable amount considering this example and also considering the survey results discussed in the following chapter.

8.1.2 Software curation

A tool for software development and curation endorsed by CERN and LHCb is the version control system GIT. Version control systems allow management of changes in software source code, documents and other collections of information. Using a version control system in analyses development allows the analysts to save the entire history of changes. This provides an ability to retrieve previous versions of every software script, which may be useful when a bug is introduced in a newer version or when results obtained with an older version need to be reproduced.

CERN has a private instance of the GIT-repository hosting service implemented with the open source software called GitLab.[1] By using an open source solution, we are able to have complete control over the system and can further develop it for our own needs. However, there are other commercial alternatives like Github [170] and Bitbucket [171].

The version control tool GIT is already widely used in the community. To make the code easier to understand and reuse, we endorse good practices, such as for example intuitive variable naming, explaining functionality with comments in the code and documentation in the GIT repository. The documentation can be implemented by creating a number of "readme" files or by using a third-party service. A README file contains information about the other files in the repository, and it is typically stored in a simple text format [172]. An example of a free and widely used third-party documentation service is READ THE DOCS [173], which also supports web-hosting and versioning of the documentation. Besides analysis code, GIT can be used for manuscripts, presentations and web pages. These files can be placed within the main analysis repository or independently in a new repository, and they also contribute to the analysis documentation.

Example 8.1.2. In the case of $D^+_{(s)} \to \pi^+\mu^+\mu^-$ analysis, its new implementation (from the previous chapter) is documented with a number of README files. The main README file is located in the root of the repository, and it presents the information like the analysis description, data provenance and technical requirements for analysis execution. The analysis workflow is sorted into three stages (preselection, multivariate selection and fitting), each of which has a

[1] It is maintained by the CERN IT group and hosted at `https://gitlab.cern.ch`.

folder in the repository. Each folder is documented with another README file that describes how to execute the stage and interpret the output.

8.1.3 Environment encapsulation

GIT provides standardised means for storing source code and build system. However, it does not provide a runtime environment that can be used to execute the software. Due to the fact that software is often a fragile component that is strictly dependent upon the compiler version, operating system or computer hardware, an old version of source code may be difficult to compile. This problem can be solved by using technologies for runtime environment encapsulation. Runtime environment encapsulation is not generally used in physics analyses at LHCb, even though it plays a vital role in software preservation.

There are various virtualisation tools that can be used for runtime environment encapsulation. These tools provide an ability to recreate an environment identical to the original and execute the code. As discussed in Chapter 5, these tools are virtual machines and containers. They are robust, reliable, and can be versioned, thus preserving the latest modifications of analysis code and workflows.

A number of DOCKER images and virtual machine images are provided by the CERN groups. The LHCb computing group provides DOCKER images that are based on SCIENTIFIC LINUX CERN 5 and 6 and CERN CENTOS 7, which can be used to capture analyses that depended on the experimental software. Furthermore, the official ROOT framework releases are now also available as DOCKER images [174]. They can be used for analyses that are performed using ROOT. If a DOCKER image is changed by analysts, who for example included additional software packages, it can be uploaded and saved at DockerHub [149] or at the CERN GitLab DOCKER registry, which is available in every repository hosted on GitLab. CernVM [175] and Micro-CernVM (introduced in Chapter 5) are virtual machine images provided by CERN, which also offer a comprehensive working environment for the HEP analysts. In addition, a number of DOCKER and VM images are provided by third-parties, which can be used as alternative solutions. Analysis preservation should not be tied to a single solution, hence there are already efforts to include images created for a new containerisation technology called SINGULARITY [176]. However, we advocate the use of the CERN provided images, as they are tailored to the needs of HEP analyses and provide a single consistent solution to the problem of runtime environment capture.

Example 8.1.3. The $D^+_{(s)} \to \pi^+\mu^+\mu^-$ analysis was developed using ROOT 6 with TMVA. This

is why, for its environment encapsulation we use a CERN provided image called `reanahub/reana-env-root6` on DockerHub [149].

8.1.4 Analysis workflows

Automatisation of the analysis workflow is a process of describing the analysis steps in a way that is convenient to use, portable and reproducible. This is typically done using a workflow engine tool, which can execute each of the analysis steps. Analysis automation can be largely beneficial for both its development and reuse, as the whole workflow could be typically set off with a single command.

In analysis workflows, we define the essence of the analysis, which serves as its documentation and allows a better understanding of the applied methods. In particular, sometimes the common practices include habits that are not sufficiently exact like for example, evaluating maxima or minima in a plot by eye, or adding contextual information while executing the code. By creating an analysis workflow, an analyst may need to improve their methods, which will decrease the chance of malpractice, and improve the quality of code.[2]

There are several ways to automate an analysis workflow. Some of them can be relatively simple, like using bash scripting, but on the other side, there are specialised, sophisticated workflow engines such as for example SNAKEMAKE [177], COMMON WORKFLOW LANGUAGE (CWL) [178] and YADAGE [179]. The choice of the tool depends on each analysis as there is a wide range of analysis at LHCb with a different number of steps and a variety of workflows. SNAKEMAKE, CWL and YADAGE workflow engines are flexible to accommodate such variety of physics analyses, and they are becoming increasingly popular in HEP analyses. They allow a portable and scalable execution of the analysis across a variety of systems, from local computers to the cloud. An example of a workflow defined by the YADAGE workflow engine is shown in Figure 8.1.

Example 8.1.4. For the automation of the $D^+_{(s)} \to \pi^+\mu^+\mu^-$ analysis, we use bash scripting inside a YAML file, which is a part of the REANA environment.[3] The analysis workflow is defined as a sequence of commands that use the ROOT framework.

[2] Quality of code is defined by its clean and understandable design and implementation, well-defined interfaces, ease of use and extensibility, availability of tests, examples and documentation.

[3] YAML (first stood for Yet Another Markup Language, then it was changed to YAML Ain't Markup Language) is a data serialisation language [181].

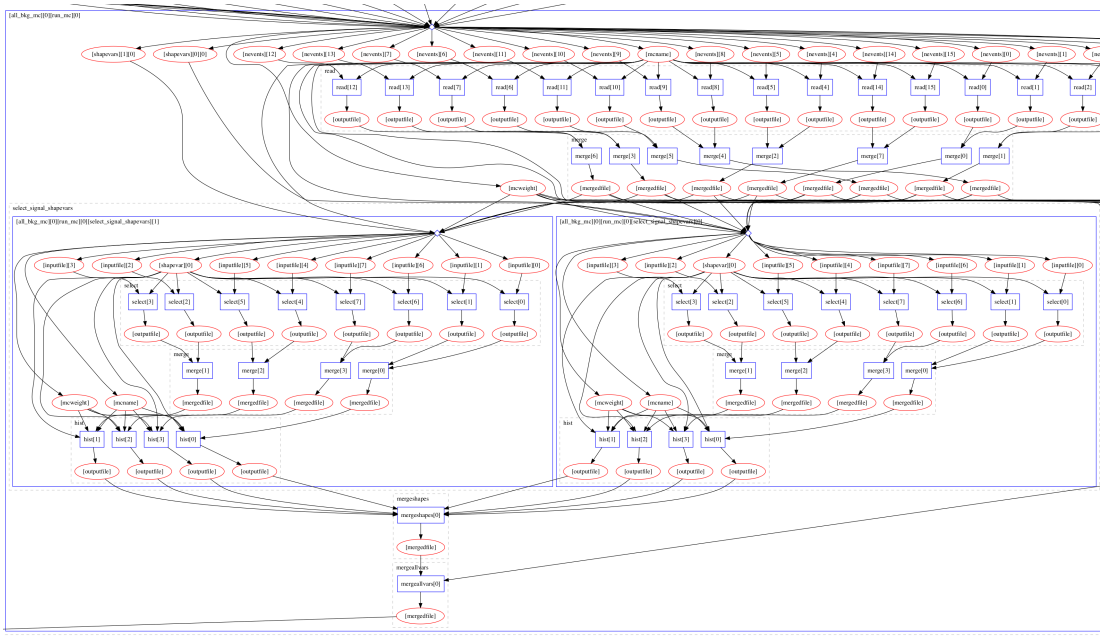


Figure 8.1: A subsection of a YADAGE workflow showing an ATLAS analysis in Beyond Standard Model searches. Processes are marked as blue rectangles, while input and output files are marked as red ellipses. Photo credit: REANA hub [180].

8.1.5 Analysis documentation

Analysis notes (ANA notes) are a standard way to document analyses methods and results at LHCb as they provide technical information on how the analysis was performed, which is typically excluded from the published papers. Additional analysis documentation is sometimes provided in the internal GIT repositories and the TWIKI [124] portal. Even when using the internal documentation, it is often not sufficient to independently reproduce analysis results (as discussed in Chapter 7). Furthermore, these resources are typically not linked to each other, making it difficult to learn what was documented. In practice, this means that often there is no reference to the GIT repository or the TWIKI page in the analysis note, even though all of these resources are internal. Likewise, often there is no reference to the GIT repository in the TWIKI. It is reasonable that these resources cannot be referenced in public papers, as they have internal access, even though this approach would be the most beneficial to further reproducibility and reuse of HEP analyses.

The failure of documentation discussed in Chapter 7 is occurring due to the current lack of formalism, where the responsibility of documentation and preservation is left to the authors. This results in excluding some of the vital information from the analysis documentation and poor referencing between the analysis resources. The pillars of analysis preservation

introduce a standard in documenting every analysis resource. Therefore, with the adoption of analysis preservation, improvement in analysis documentation will naturally follow. I discuss this transition and its implications in Chapter 9.3.

8.2 Collaborative work

Collaborative working covers a variety of ways in which groups and collaborations can work together. The LHCb collaboration currently has eight physics working groups[4]. Each group addresses particular physics phenomena. For example, there is a group working on solely rare decays or another looking into charm physics. However, there is often overlap in the groups, meaning that one researcher can contribute to several groups. This implies that the groups need to work together, but they also need to organise their own research within the group. Tools for collaborative working can help the group achieve their scientific goals more effectively and efficiently. The current examples of the collaborative working tools at LHCb are the common storage space on the EOS system and the real-time communication service with MATTERMOST [182].

The tools used for preservation are strongly linked to collaborative working, and as such, they can immediately support and help analysis development. This is manifested in several different scenarios described in the following sections.

8.2.1 Software development

Software developers usually work on different tasks within a software project, and their contributions need to be integrated at a later stage of the development. The situation is similar with physics analyses in HEP, as within a physics working group, the workload is usually divided among the analysts who need to solve specific tasks by writing analysis code.

In collaborative work when the code is written by many contributors, there is a constant risk of introducing new bugs or inconsistencies between the components. This can be solved by using a system called continuous integration (CI). CI is a process of automatisation of the software build and testing, which emerged in the late 1990s to be one of the most widely used practices in software development and preservation. This approach is similarly starting to find its use in the applied sciences. It encourages code sharing and small contributions which can be transparently tested within the whole code structure. The CI server monitors changes in the

[4]LHCb working group database: `lhcb-wg.web.cern.ch`

source code, and with each new commit, it triggers the build of the code. The code is then validated and tested, after which the server sends a notification whether the build succeeded or failed. The schema of this procedure is shown in Figure 8.2. Using a version control system with CI can be largely beneficial for the analysis code development because bugs in the code are more likely to be caught early and thus are often easier to solve.

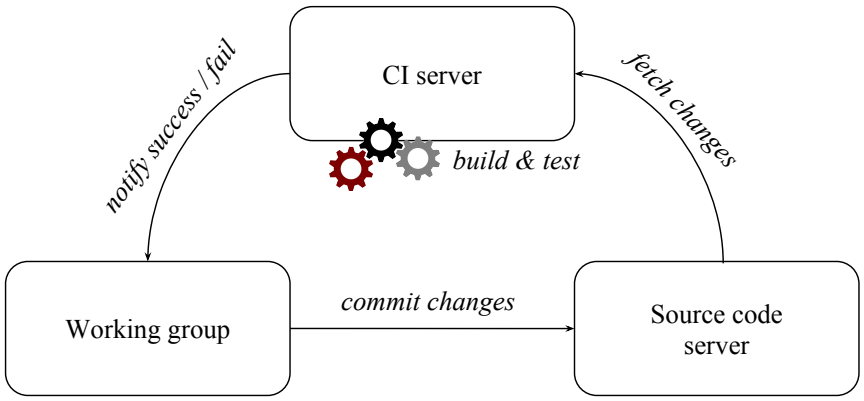


Figure 8.2: Continuous integration procedure.

Even though CI is used for the LHCb experimental software, it is rarely used for software development in physics analyses. The CERN GitLab instance provides an integrated CI system, and the analysts should leverage it to test their contribution before it is integrated into the main analysis repository. Well-tested and robust software is easier to preserve.

8.2.2 Review and project organisation

Once software developers want to submit their contributions to the code repository, they make a "merge request" to the repository administrators who can merge the changes. The *fork-merge* approach is advised when performing physics analysis because the code is being reviewed multiple times, the first time by the original author and the second time by the administrators of the repository.

The CERN GitLab instance provides a user-friendly graphical interface, which can facilitate the code review process. In addition, it provides issue tracking where each member can file an issue explaining a potential problem in the code. Finally, the integrated tool called JIRA [183] provides organisation functionality with schedules and milestones for the community. It is commonly used at LHCb for the development of the experimental software. These resources amend the existing documentation and help with software preservation in the long term.

8.2.3 Education and training

In the last decade computational and programming skills have become increasingly relevant for data analysis in HEP. When novice analysts join HEP collaborations, they need to learn programming, data analysis and statistics to be able to study the collision data effectively. Physics analyses often include developing mathematical models, writing algorithms and using advanced computing systems such as the LHC Worldwide Grid. This is why software tutorials and workshops called the LHCb Starterkit [184] have been organised to train new members of the collaboration. The program of the workshop is frequently updated to include new software features and, even though it already promotes best practices in software development, it can also promote analysis preservation tools. Furthermore, previously preserved analyses can be a useful educational resource for someone who has recently joined the collaboration.

8.3 The CERN Analysis Preservation framework

The CERN Analysis Preservation framework, as introduced in Chapter 1.7.2 provides a set of tools for analysis preservation and reuse in HEP. These are in particular the analysis form, CAP client [185] and the computing infrastructure provided by the REANA [186] project. It is a unique framework, and there are currently no other similar alternatives for data analysis preservation.

8.3.1 The analysis form

The form to document the LHCb physics analyses has been changing and evolving in the previous years. The idea is that it should be as informative as possible, while still not burdening the authors with providing inaccessible information about the analysis.[5] Together with a few other representatives from LHCb, I collaborated with the team of developers who implemented the CAP form to make sure that it captures comprehensive information about LHCb physics analysis.

Each of the LHC experiments has a specially designed form, which is customised to their requirements. With the analysis form, the service imposes a standardised way to document analysis and thus facilitate reproducibility. The form is divided into several sections that include basic information, data provenance, analysis code and additional resources such as presentations

[5] These inaccessible information could be about data or software, but to obtain them additional knowledge may be required.

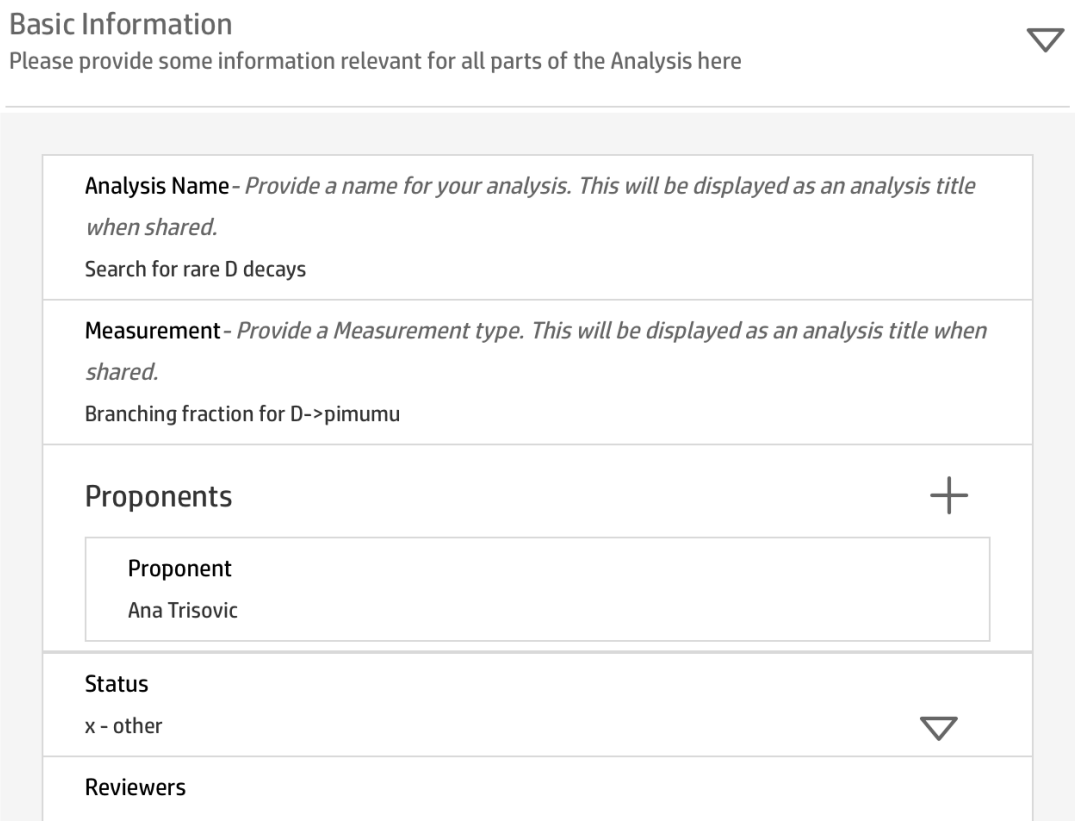


Figure 8.3: Basic information in the CAP form.

and publications.

Example 8.3.1. Using the $D^+_{(s)} \to \pi^+\mu^+\mu^-$ analysis as an example, I demonstrate the usage and flexibility of the CAP form.

The first section of the form includes the basic details. Besides the analysis name and measurement as shown in Figure 8.3, there are text fields to specify the proponents, status of the analysis, reviewers and working groups. Completing the form, I named the analysis "Search for rare D decays" and the measurement "Branching fraction for D–>pimumu". There are several options for the status of the analysis like for instance "in preparation" or "journal review", but I chose "other", as this is an analysis carried out solely for the purposes of testing reproducibility.

The second section of the form addresses data provenance or "Stripping and Turbo selections" of the input data. In my analysis, I use eight different datasets. Four of them are real data collected under different conditions, in 2011 and 2012 with magnet up and down in each year (as documented in Figure 8.4). The other four are Monte Carlo samples that mirror the data conditions. The data provenance is uniquely linked to the LHCb bookkeeping paths, which are fairly descriptive (as discussed in Chapter 4) and therefore very valuable for analysis preservation and reuse. The form allows adding a large number of different datasets (which can

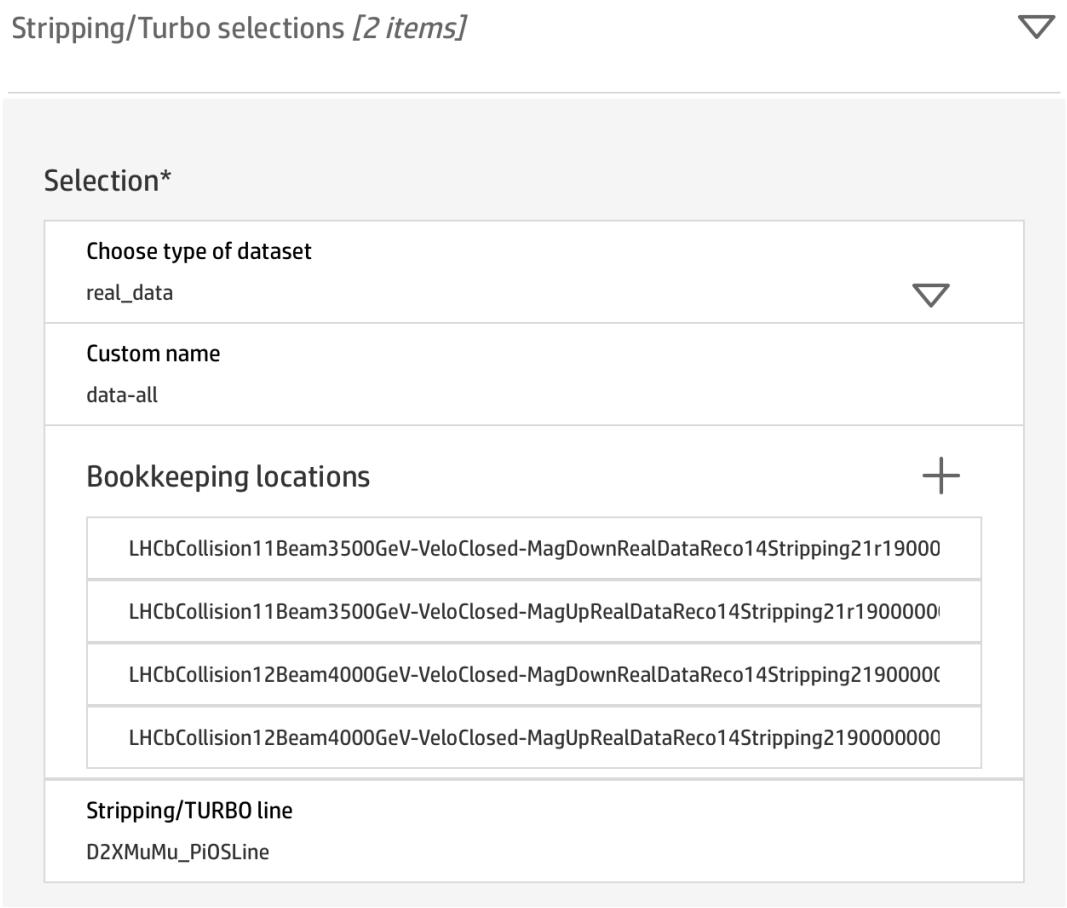


Figure 8.4: Data provenance in the CAP form.

be achieved by clicking the button “+”).

Any change in the form will be reflected in the JSON schema. These schemes provide flexible metadata structures that describe the analyses. Therefore, in the backend, the data provenance section is captured as the following:

```
"stripping_turbo_selection": [
  {
      "bookkeping_locations": [
          "LHCbCollision11Beam3500GeV-VeloClosed-MagDown
              RealDataReco14Stripping21r1
             90000000CHARMMDST",
          "LHCbCollision11Beam3500GeV-VeloClosed-MagUp
             RealDataReco14Stripping21r1
             90000000CHARMMDST",
          "LHCbCollision12Beam4000GeV-VeloClosed-MagDown
             RealDataReco14Stripping21
             90000000CHARMMDST",
          "LHCbCollision12Beam4000GeV-VeloClosed-MagUp
             RealDataReco14Stripping21
             90000000CHARMMDST"
      ]
      "dataset_type": "real_data",
      "name": "data-all",
      "stripping_turbo_line": "D2XMuMu_PiOSLine"
  }
  ...
]
```

In order to have a high-level of flexibility in regards to capturing analysis software, the whole analysis GIT repository is documented and copied via the form. The form allows the preservation of one or more repositories per analysis. In addition, it provides a functionality to copy the DOCKER containers from the GitLab registry or DockerHub [149]. These form fields are shown in Figure 8.5.

Finally, the form documents internal discussions, presentations, publications and other

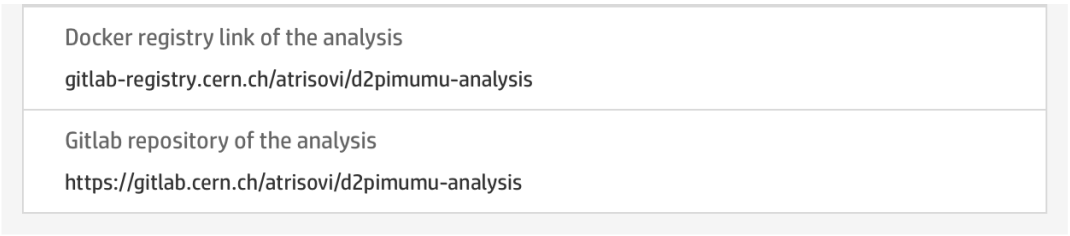


Figure 8.5: Once the URL is specified in the form, the entire repository is fetched, versioned and archived in CAP.

documentation in the last section, shown in Figure 8.6.

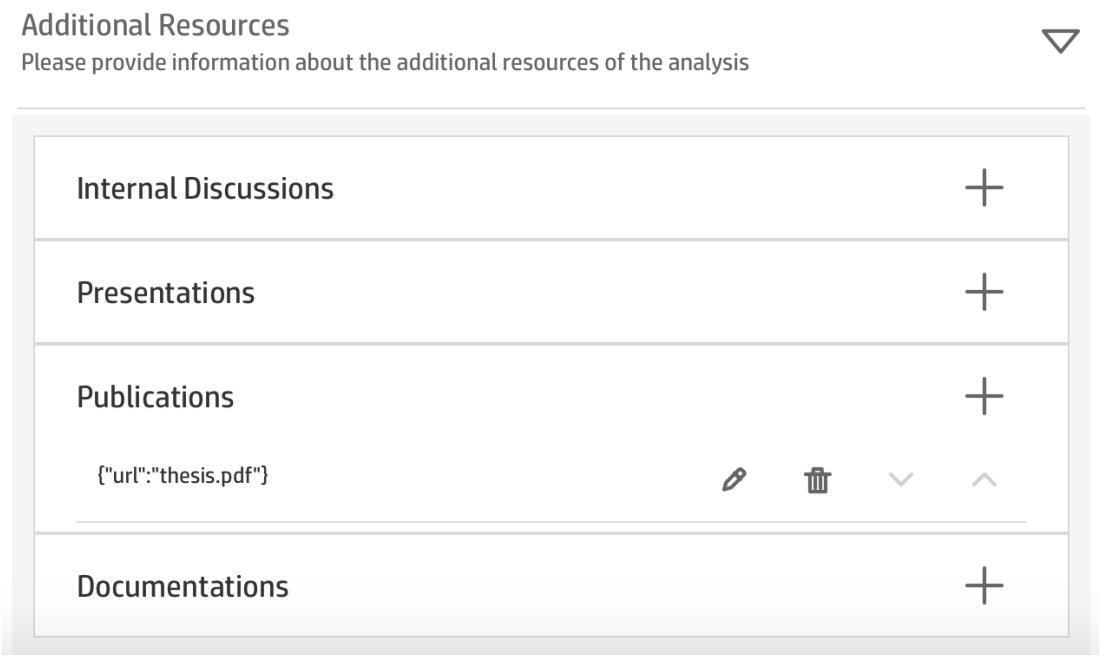


Figure 8.6: Documentation and publication section in the CAP form.

8.3.2 Preservation of the analysis software and Docker images

I have worked closely with the CAP development team and had an important role in the implementation of CAP's integration with GIT [187]. In particular, the goal was to preserve the code and DOCKER images that are hosted in GitLab, GitHub or DockerHub repositories.

Once the analysis code is completed or reached a release version, it is a good practice to version it with a label, such as for example `v1.0`. This release version can then be archived as the source code in the CAP framework. The CAP server imports only the final version of the code without copying the entire history of changes. However, as the code and DOCKER images can be versioned, each released version is stored on the framework.

For the GIT integration, we use two python packages that provide an access to the GitLab and GitHub server API, and those are PYTHON-GITLAB [188] and PYGITHUB [189] respectively. For the preservation of DOCKER images, we chose the command-line utility that performs various operations on image repositories called SKOPEO [190], which is used by the CAP server to copy and store images from either the GitLab or DockerHub registry.

8.3.3 The CAP client

CAP client is a command-line tool developed within the CAP framework to facilitate preservation of the analysis resources. It is developed as a free and open source python package. Its functions currently include:

- `create`, `clone` and `delete` analyses,
- `edit` the analysis schema,
- `upload` and management of files (like datasets),
- `publish` and management of permissions of the analysis.

One of the features I contributed to was enabling the management of analysis metadata. Information about an analysis is stored as a dictionary in a JSON schema, meaning that its details can be accessed by specifying the keys of the dictionary. The part of the schema that is completed by the analysts is called “metadata”, while other automatic parts include entries like, for example, date of completion. With the new feature to manage the metadata, analysts can update their analysis record directly from the command line. An example of this functionality is shown in Figure 8.7.

The CAP client has great potential in the automatic preservation of software within the GitLab CI system. In particular, once a new version of the software is released, the client can automatically archive the repository and send it to the CAP server for preservation as a part of the CI workflow.

8.3.4 Reusable analysis

The REANA project, introduced in Chapter 5.5, is a service dedicated to instantiating physics analyses using technologies like Docker and KUBERNETES. It leverages the preserved resources such as analysis code, computational environment and workflows to execute the analysis on the CERN cloud. The current plan of the project is to enable integration between the

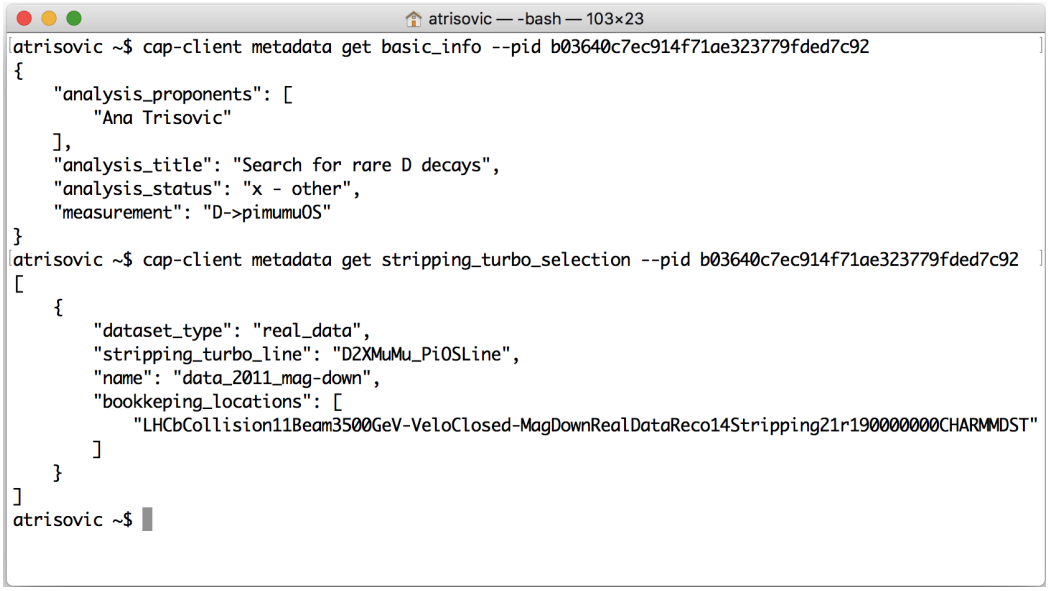


Figure 8.7: Metadata functionality in the CAP client. `pid` is a unique identifier of every analysis in the CAP framework.

CAP form and the REANA infrastructure, thus allowing any preserved analysis on CAP to be effortlessly executed in REANA (ideally with a button click).

The $D^+_{(s)} \to \pi^+\mu^+\mu^-$ analysis is featuring as a working example at the REANA hub. Its reproducibility was tested by the members of the CAP team, who successfully reproduced the plots. Any member of CERN can rerun and reuse this analysis.

8.4 Chapter summary

The analysis preservation pillars define a standard in analysis documentation and preservation. The standard covers the preservation of the data and documentation, and curation of the code, runtime environment and workflows. The tools used for preservation can immediately improve performance and collaborative work of active analyses, and as such should be promoted in the community.

The CAP knowledge preserving framework has a crucial role in analysis preservation and reusability. It is the CERN supported service that implements the preservation of the main analysis pillars. Using the CAP form, python client and the REANA infrastructure, we cover every aspect of preserving physics analyses and thus facilitate reproducibility.

In the following chapter, I present a survey that was conducted among the analysts to learn more about common analysis practice. I discuss barriers to the adoption of analysis preservation by examining different sociological aspects.

Chapter 9

Retrospect on scientific preservation

In the previous chapter, we learnt about the best practices in performing physics analyses and how they are conducive to analysis preservation. However, using the analysis preservation tools often require additional training and expertise in computer science. We need to assess whether the HEP community is receptive to adopting these tools and identify potential barriers.

I conducted a survey to learn how familiar the analysts are with various analysis preservation tools. In this chapter, I present the results and deduce implications of this survey. Finally, I comment on how the current collaborative tools are used in the HEP community and consider sociological aspects of adopting analysis preservation.

9.1 Survey on physics analyses performance

In order to assess the status of analysis preservation and learn about current analysis practice, we conducted a survey among the HEP analysts [191]. Having initially received a relatively small set of responses, the survey was conducted one more time to increase the sample and improve the study.[1] The first time it was conducted was in April 2017 during the first analysis preservation "hackathon" at LHCb. The participants were mostly PhD candidates in physics, but there were also a few team leaders who were mentoring analyses. This set of 14 responses was later labelled as "computing enthusiasts" in Ref. [191] since the people who attended the hackathon may not be representative of analysts in general.

The survey was repeated in March 2018. This introduced additional 24 entries consisting entirely of graduate students. Their responses represent an "average" graduate student skill-set,

[1] We made sure not to include the same person two times in the survey.

and they were labelled as “students” in Ref. [191]. The survey was predominantly completed by the LHCb analysts, however, there were few contributors coming from the other LHC experiments. The survey results revealed information on common analysis practice, code sharing, curation and the use of tools conducive to analysis preservation, as we will see in the following.

9.1.1 Common analysis practice

Common analysis practice signifies how analyses are normally done in the collaboration. Analysis preservation should not impose radical changes in the common practices, but be sympathetic to them. In order to learn more about the ways analyses are performed, we asked the participants questions like “How often do they reproduce their ntuples?”, “How big are the ntuples typically?”, “Do they use the CERN infrastructure?”.

To the question of how often the analysts reproduce their ntuples, 60.5% of the participants stated ‘every few months’, 21.1% stated ‘every few weeks’ and 13.2% stated that this was done ‘only once’. The results are shown in Figure 9.1. The variety of responses is probably caused by the different stages of analysis development, i.e. if the analysis is only initiated the ntuples would have been created only once. However, in most of the cases, ntuples are commonly reproduced multiple times in analysis development. The fact that one is often repeating this process is an argument for analysis automation.

Another question was: “how can your analysis results be reproduced?”. Some of the participants (less than 20%) indicated that their analysis is automated and can be reproduced with a single command that sets off the data processing. Around 60% said that their analysis could be reproduced following a series of steps that, in the end, produce their results. Finally, 20% of the participants stated that they depend on their collaborators to produce some intermediate result in the analysis workflow. With several people having an automated analysis workflow with the collection of analysis resources in one place, we can hope that the community is heading in the direction of higher automation and preservation. However, we observe cases where analysts are dependent on a coworker to produce an intermediate result. This dispersal of analysis resources can reduce productivity and hinder reproducibility.

When it comes to the ntuple size, the sizes of the datasets are said to range from 1 GB to 600 GB, averaging to 93 GB per analysis. The wide range of reported sizes may exist due to different sorts of analysis. Also, it may be due to different stages in the analysis process, meaning that just initiated analyses work with ‘large’ datasets while finalising analyses work

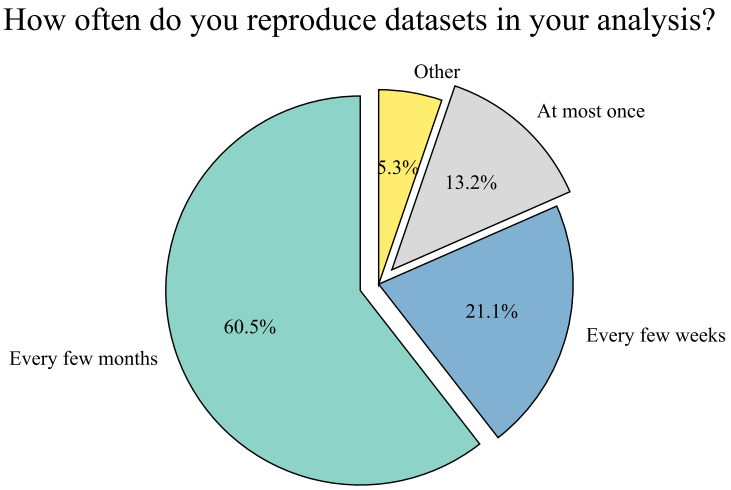


Figure 9.1: Survey results on question: "How often do you reproduce final datasets in your analysis?"

with 'small' datasets. Because of this, the results were not very informative, so another survey was conducted to only learn more about the average data size. The new replies included entries by analysis working groups, who work with datasets recorded in Run 1. From the replies we find that the input dataset on average include 140 GB of real data, approximately 40 GB of Monte Carlo simulation and approximately 70 GB of other data (which is presumably calibration data used to find systematic errors). The results are shown in Figure 9.2. In total, this is on average 250 GB per analysis.

This means that for analysis preservation we likely need to allocate approximately 300 GB of storage per analysis performed on Run 1 data to include input data and other resources. As mentioned in Chapter 8, the CAP framework currently provides up to 1 TB of storage per analysis for the preservation of input data. Given the responses we received, this amount sounds reasonable. However, we expect that the storage requirements will increase in the future with the volume of the data.

One of the questions addressed what computing infrastructure the analysts typically use for performing their physics analyses. Here we learn that around 70% of the participants use their own laptops or desktop computers to fully or in part perform their analysis. All of the participants indicated that they to some degree use a local university cluster, shared CERN infrastructure via LXPLUS and the Grid. Some of the participants (around 30%) use the CERN cloud resources for their analysis. Responses were not mutually exclusive, and all of the

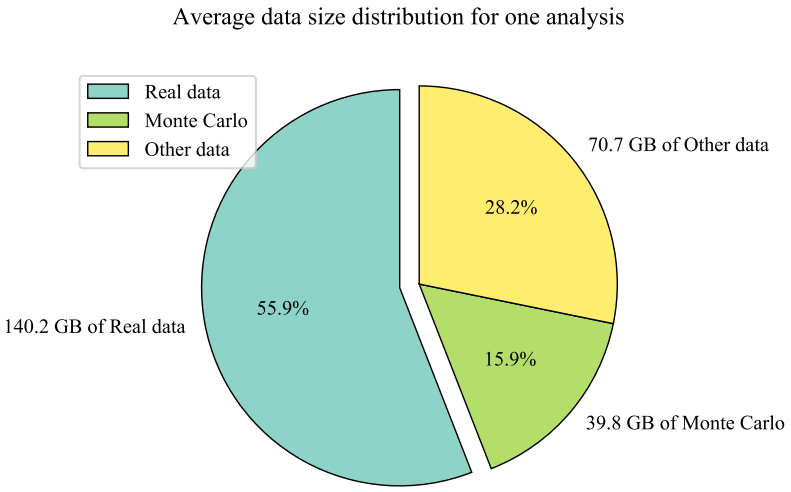


Figure 9.2: Survey results on question: "What are the input data sizes in your analysis?" The answers were averaged out from 16 different analysis.

participants use at least two of these computing resources.

Here we learnt that the analysts use a wide range of computing infrastructure and that analysis preservation should not be tied to a single solution. However, we note that the analysts work mostly on a Linux-based operating system (LXPLUS, CERN cloud and university clusters), meaning that Linux-based containers and VMs are well-suited for analysis preservation.

9.1.2 Code sharing and curation

Following the survey section on analysis common practice, we inquired about analysis software (code) curation and its development. The majority of the analysts (around 84%) are GIT users, and they have analysis repositories on either GitHub or GitLab (sometimes both). Around 8% of the analysts keep the code on Subversion (SVN), while the other 8% do not have a code repository in a version control system. These replies are presented in Figure 9.3.

The fact that most of the analysts use either GIT or SVN represents a good starting point for analysis and software preservation. This is particularly significant because the use of a version control system is not strictly required, yet it is recognised as a beneficial and effective tool in the community.

Regarding code sharing, 63.9% of the analysts indicated that they have access to their colleagues' code. Around 30% said that they have 'access on request' and around 5% said that they 'do not have access'. These results are shown in Figure 9.4. Considering that from 64% to

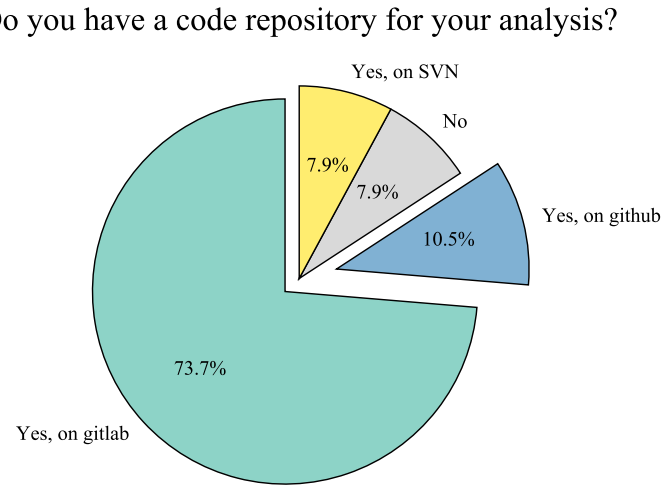


Figure 9.3: Survey questions on code curation.

94% of the analysts share code is a positive indication of collaborative work and software reuse.

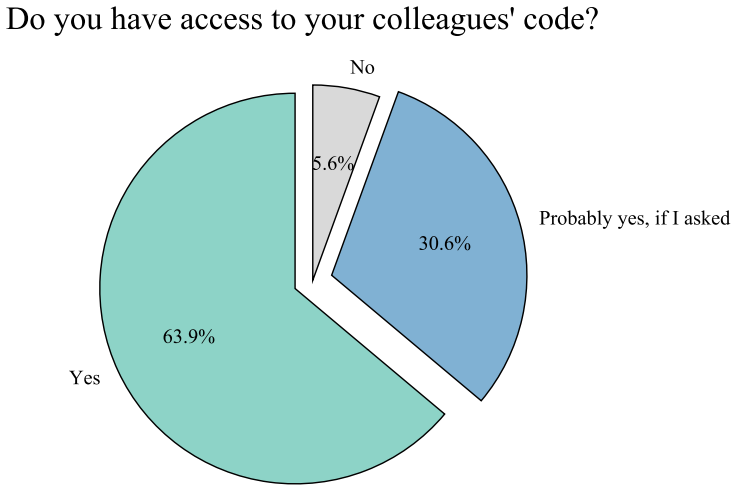


Figure 9.4: Survey questions on code sharing.

9.1.3 Analysis preservation tools

Many analysis preservation tools require some understanding of software engineering. Some of the particularly helpful tools for analysis preservation are the cloud computing software CERN OPEN STACK, the virtualisation technology DOCKER, workflow systems and continuous integration. These are described in detail in Chapter 5 and Chapter 8. We asked the analysts

how familiar they are with the tools and received the results shown in Figure 9.5.

More than half of the participants have never used either CERN OPEN STACK, analysis workflow systems or continuous integration. Almost two-thirds of the participants, 78.9%, have never used DOCKER . There are only 5.2% of the analysts who comfortably use CERN OPEN STACK and DOCKER , and there are about 25% of the people who work with analysis workflows and continuous integration.

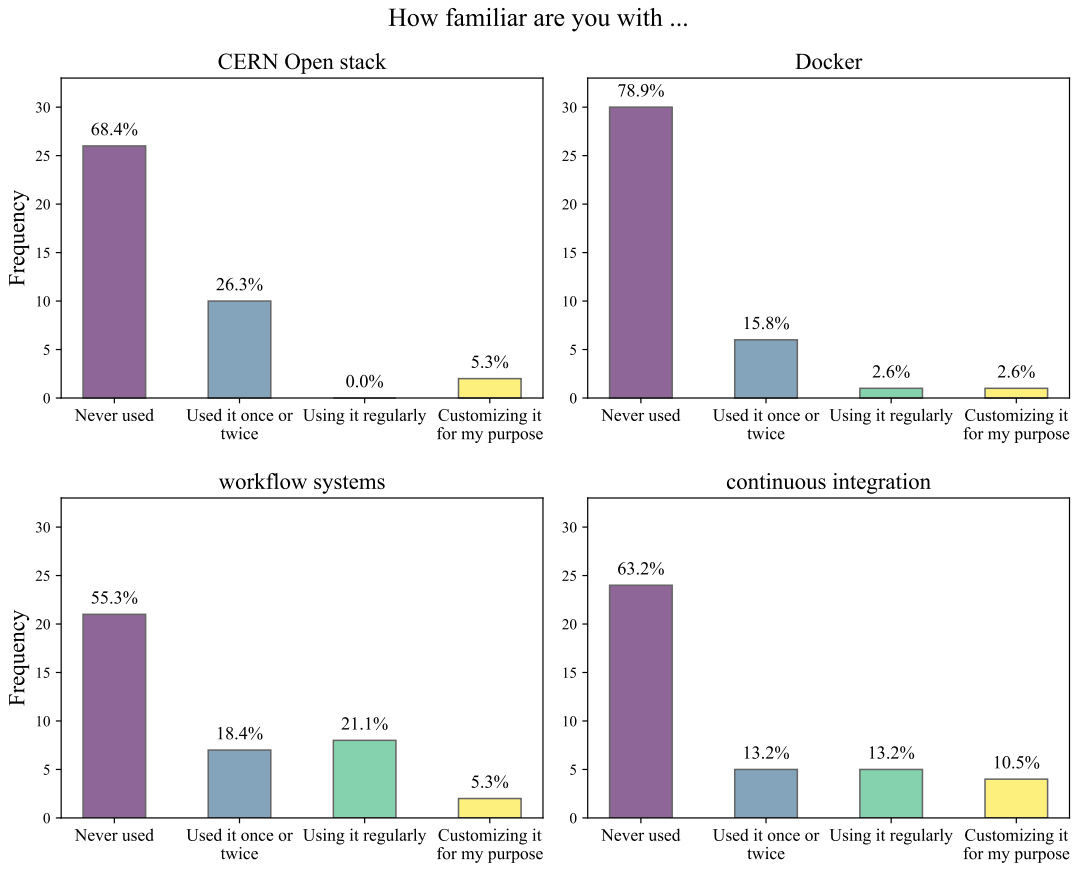


Figure 9.5: Survey results to the question how familiar the users are with different tools used for analysis preservation.

This outcome was somewhat expected as these tools are not strictly required in the analysis development. However, the fact that they are partly in use is an indication that some analysts are interested in improving their workflow and methods by employing ‘auxiliary’ tools.

Regarding the difference in two surveys (one labelled ‘students’ and the other ‘computing enthusiasts’), I note that the ‘computing enthusiasts’ are more comfortable using the analysis preservation tools, even though the majority of these replies also indicate that the tools were never used. The comparison between the results for computing enthusiasts and students is presented in the repository in Ref. [191].

In the analysis preservation initiative, the use of all of these tools is advocated. However, the survey results show that the community is not yet fully comfortable to employ all of them. This means that the introduction of new tools via training and online manuals needs to be gradual.

The choice of software applications and packages vary between the analyses. To learn more about that, one of the questions in the survey was "what software do the analysts mostly use?". According to the variety of responses, this question was potentially interpreted in a number of different ways. The analysts listed tools for building the software and also programming languages that they use to write code and process the data.

Most of the participants use python, ROOT and C++ as shown in Figure 9.6, which was expected as these tools are typically used for LHCb physics analysis. The official experimental software seems to be used by 52.6% of the analysts. However, LHCb data can only be retrieved using the experimental software, and the fact that this is not 100% indicates that many analysts use already created ROOT ntuples. For instance, the ntuples may have been created by their colleague as part of collaborative work. In addition, the lack of responses about the experimental software may be caused as in the first survey the analysts listed the tools themselves (there were no options to choose from), and they may have omitted to mention experimental software as a part of their workflow.

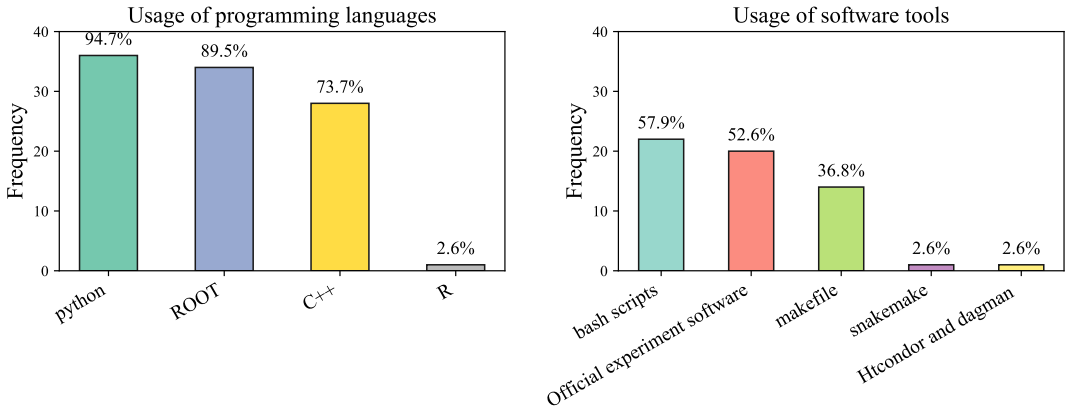


Figure 9.6: Survey results on question: "Which software do you use in your analysis?"

Regarding software build and execution, half of the participants use BASH SCRIPTS, which is typical for Linux-based operating systems. Also, the build automation tool MAKEFILE [192] is often used in the community with 36.8% of the entries. Other auxiliary tools include the workflow management system SNAKEMAKE [193], the programming language for statistical computing R [194] and high-throughput computing software HTCONDOR with

DAGMAN [195]. The use of such other tools is unconstrained, meaning that analysts can use whatever they find necessary, and if there were more participants in the survey, we expect to have seen a large variety of alternative projects.

It is important to note that most analysts are comfortable working in python and its environment. Thus, the survey results indicate that the python tools may be more easily adopted in the community. This is a positive implication since several analysis preservation tools (such as for example the CAP client [185]) are currently being developed at CERN using python. Also, many of the endorsed tools (like workflow engines) are implemented as python packages.

9.1.4 Survey implications

The survey discussed in the previous section captures responses from a small number of people compared to the sizes of the HEP collaborations. Even though our statistics are low due to the small turnout, we are able to draw some general conclusions. Notably, the survey included representatives from 8 out of 8 different LHCb analysis working groups, which gives us a flavour of analysis preservation efforts within the groups.

Studying the survey results, we conclude that there is a broad analysis diversity, both in terms of its complexity and analysts' working experience. Therefore there seems not to exist a uniform solution to fit all, but the solutions need to be flexible to meet the needs of each analysis.

We observe that the actual analysts' computational expertise may be lower than expected by the analysis preservation initiative. This means that going forward preservation tools need to become more comfortable to use, but also adequate training for the analysts needs to be provided. A strategy of training and streamlining of tool usage is necessary to increase their employment in physics analyses.

9.2 HEP and collaborative tools

Over the years we have seen examples of successful and failed collaborative tools in HEP. An example of a hugely successful tool is the ROOT forum[2]. This forum allows any user to post their problems and questions about using ROOT. The forum does a tremendous job in focusing the experts' attention by classifying the users' posts. The experts reply to the posts, often solving these problems within minutes, while the original author of the post might take much longer to solve them on their own. The forum activity is high, with tens of new posts

[2]ROOT Forum: `https://root-forum.cern.ch`

emerging every day. Furthermore, it serves as additional documentation of the ROOT framework as the users can search through its history.

In LHCb, there was a similar idea to support the collaboration software users with the LHCb Q&A portal [126]. The portal also allows classification of the posts to draw the attention of the experts. For example, DAVINCI developers might look at posts only labelled with the "davinci" tag. However, probably due to the lack of advertisement and a smaller number of users, the portal is not as active as the ROOT forum. Hence, for a collaborative tool to be useful, a significant effort must be put into advertising and creating incentives for its use. In the case of LHCb Q&A, users often turn to the internal mailing lists to look for help, meaning that a competing alternative already exists, thus making it challenging to introduce a new tool, even though it might be better.

In the recent years, we have seen a rise of several new tools, such as the CERN Analysis Preservation (CAP) portal, LHCb Q&A, Swan and Everware (introduced in Chapter 1.7.4). Many of them are developed at CERN and tailored to the needs of the HEP community. These young projects have high potentials to enhance collaborative work. In particular, allowing communication of results through Swan or HEPdata and information sharing though CAP or LHCb Q&A improves chances of reproducibility and reuse. However, due to insufficient training or interest, there is a real danger that these projects will fail. Even if the marginal benefit of preservation tools for a single researcher may seem small, there is a tangible benefit in the long run, not only to the individual analysts but to the HEP community in general. We recognise that many of the barriers to adopting new tools are sociological, some of which we explore in the following section.

9.3 Sociological aspects

Exploring the sociological aspects affecting the adoption of new ideas is essential to understanding whether they are going to succeed or fail. Here, I outline three situations of friction in the adoption of new methods in collaborative work, which can impose a hurdle to the current and future preservation efforts.

The first and most common argument is that "new tools are difficult to learn and present a burden to already preoccupied analysts". The tools might be initially taxing to learn, but they are very beneficial in the long term. Additionally, the earlier they are learnt, the more of a benefit they will provide. We can facilitate this process by being committed to supporting

the analysts in making the process of learning more accessible. Furthermore, for the DOCKER technology, there is a HEP software initiative that provides stable images for the analysts to work on [196]. The analysts do not need to start from scratch but can use an existing solution base. An excellent example of the widespread adoption of a relatively complicated (but immensely useful) software is GIT . In the previous years, a version control system was nearly completely adopted in analysis development.

Secondly, we recognise that analysts are burdened with a pressing demand to publish papers and present physics results [167]. This is due to the fact that young scientists, who conduct most of the physics analyses, have typically fixed-term work arrangements with the collaboration. They need to focus on their physics results to complete their studies or ensure future jobs. However, this imposes a problem in regards to the analysis preservation. Analysts are often enthusiastic about the development of an analysis preservation tool. However, when they are asked to use the tools, this enthusiasm often does not match the willingness to preserve. They hope that someone else will take the lead, or that they could contribute at a later stage. By that time it is often too late because the project moved on or its developers left the group. This is possibly what caused disinterest in the LHCb Q&A portal and it may also happen with the analysis preservation tools.

Finally, people are afraid of their methods being scrutinised. This is the reason why the analysis software is not often public, and when it is, it may be written in an incomprehensible way. Furthermore, people are uncomfortable exposing flaws in their colleagues work, and this is one of the reasons why code review is not a favoured task. However, if code review is required, then analysts would need to do it, giving valuable feedback to their peers and possibly exposing incorrect methods and bugs. We do however realise that sharing and review of code are hard to impose without policies implemented at the collaboration level.

9.4 Chapter summary

At the beginning of this project in 2014, it was clear that there were many topics and issues related to research preservation that needed to be addressed. Data preservation was not a collaboration priority in LHCb and there was not much discussion about it. Since then, the situation has changed and nowadays there is usually a session dedicated to data and analysis preservation at many conferences and meetings. Specialised workshops such as the LHCb analysis preservation hackathon, which focuses on hands-on tutorials and tools facilitating

preservation, have been organised. The same trend is followed by science and technology in general and topics on data preservation and reproducible research have been more and more present.

The survey introduced in this chapter gave us a better overview of how analyses are done and allowed us to set future goals for the analysis preservation initiative. From the survey results, we learn that our community at present is not ready for full analysis preservation but there are hints of a coming transition, with the widespread use of GIT as one of the strongest indicators.

We also learnt that the community (mostly) has access to the code repositories by their coworkers. In some cases, these repositories are even made public and accessible to everyone. These are the first but valuable steps towards Open Science. In the following chapter, I summarise the findings of this thesis and give an outlook on the future.

Chapter 10

Summary and outlook

This thesis presented the first study of data preservation and research reproducibility at the Large Hadron Collider at CERN. These topics are of increasing relevance due to the rising complexity of physics analysis, the rapid expansion of the volumes of collected data and increasing requirements of research data management from the funding bodies and scientific journals.

I have investigated how specialised tools and practices can facilitate data preservation and analysis reproducibility in the context of the LHCb experiment and in the context of the LHC in general. In particular, I have developed software aimed at capturing data provenance and facilitating analysis reproducibility. I have studied existing tools and methods which can be employed to aid reproducibility, and I have also assessed the challenges related to reproducing an LHCb physics analysis and suggested improvements that can help one reach the goal of full reproducibility. In the following, I summarise the obtained results and give an outlook on possible further work.

10.1 Summary

In Chapters 2 and 4 we studied the flow of data from the raw output of the detectors to the final results presented in physics publications at the LHCb experiment. We found that the link between a dataset and the processing steps used to obtain that dataset was obscure and not readily accessible to the analysts. In order to solve the issue, which is a clear obstacle to analysis reproducibility, a graph-based provenance database was designed and developed. The database contains three types of nodes: datasets with their processing information, software

application and version, and hardware architecture that the software is compatible with. The edges in the graph represent dependencies among the nodes, i.e. the dependencies of a produced datasets of certain software and the dependencies of the required software of certain hardware. The graph is built on top of the NEO4J database technology, and it provides a central system to preserve information that was previously scattered across the LHCb computing infrastructure. Furthermore, the database can be used to deduce which software applications and hardware was used the most and the least, thus providing a natural way to prioritise which older application versions and hardware should be readily accessible in the future.

In Chapter 5, by using the implemented graph database, a methodology to recreate the LHCb computational environment and to execute the data processing on the cloud was developed and implemented. The implementation was based on virtual machines and DOCKER containers. The methodology was thoroughly tested with Monte Carlo simulations from Run 1, and when datasets were recreated on the cloud it was found that the produced physics events were identical to the ones in the official LHCb data. This shows that we are able to mimic the original computational environment to generate LHCb datasets or Monte Carlo simulations with third-party software. This system thus provides a streamlined way to recreate datasets that depend on potentially obsolete LHCb software at the time that the dataset needs to be recreated. Furthermore, this system is convenient for purposes of outreach as it can provide means to process LHCb open data to citizens and physicists that are not part of the LHCb collaboration. This is possible due to the fact that the official experimental software is free and open access and that there are various publicly available clouds that can be used for this purpose.

In Chapter 6 we turn to the reproducibility of LHCb physics analyses. A data provenance tracking service was implemented within the LHCb software framework GAUDI. This service allows analysts to capture the data processing configurations used to generate their datasets within the datasets themselves. Bundling this metadata together with the datasets reduces the probability that the information about the processing steps is lost and makes the information available, thus facilitating reproducibility.

To assess the challenges related to reproducing an LHCb physics analysis, in Chapter 7 we attempted to reproduce the major parts of the analysis presented at Ref. [158]. For this exercise, only publicly and internally available documentation was utilised. This study allowed the identification of barriers to reproducibility and specific points where documentation is lacking. With this knowledge, one can target specific areas where improvement is needed and where different or improved practices should be encouraged in the future. Concretely, we note

that the use of a version control system, a workflow system and environment virtualisation can significantly improve the reproducibility of an analysis.

In Chapter 8 we introduced and extended on existing tools for research reproducibility. We presented in detail the CERN Analysis Preservation framework (CAP), which is a general knowledge preservation framework developed at CERN to be used across all the LHC experiments. The functionality to preserve source code from GIT repositories and DOCKER images was one of the contributions that was integrated into the framework.

Finally, in Chapter 9 we carry out a study to learn about some of the sociological aspects of the adoption of analysis preservation. We note that in the last years the experiments have come together to collaborate on scientific preservation and they have created study groups dedicated to these issues. Furthermore, new meetings strictly focused on analysis preservation have been organised at LHCb in the last years. A questionnaire that was completed by almost 40 analysts at LHCb has been presented to learn about the individual analyst's attitude towards various practices and knowledge of software useful for reproducibility efforts. We find that tools that aid reproducibility are becoming increasingly utilised. However, we also see that majority of the people are not familiar with the tools, implying that a training program for these tools needs to be envisaged.

In conclusion, it is important to note that many of the observed obstacles to reaching full reproducibility can be surmounted by the use of best practices and specialised software tools, which are also conducive to ongoing physics analyses.

10.2 Outlook

Every year approximately sixty physics analyses are published by LHCb, with the exact numbers shown in Figure 10.1. However, few of these analyses are preserved or easily reproducible. This occurs for a number of reasons that were previously mentioned in this thesis. Firstly, many of the analysts have short-term arrangements with the collaboration to work on a particular analysis, during which they need to focus on physics results and publications. Secondly, many of them are not aware of the analysis preservation efforts taking place at CERN, and it seems likely that a larger number of people would perform their analyses with reproducibility in mind if they knew about these efforts. Thirdly, analysis preservation is not well incentivised, unlike the production of physics publications. Finally, analysis preservation and reproducibility is usually not encouraged by supervisors or physics conveners.

In order to improve the current situation, a number of efforts are taking place at LHCb to promote and raise awareness about analysis preservation. An unofficial working group has emerged and released a document to outline the LHCb roadmap for analysis preservation in Ref. [197]. The document collects a number of best practices and tools that can be used for analysis preservation, and some of the recommended tools are already being actively employed. Furthermore, the CERN Analysis Preservation framework has more features than ever before, and the development group currently counts a record number of contributors. It appears that there is more interest in analysis preservation than before within the community.

Considering the work presented in this thesis, there are several natural ways in which it could be extended, and in the following, I present some possible studies.

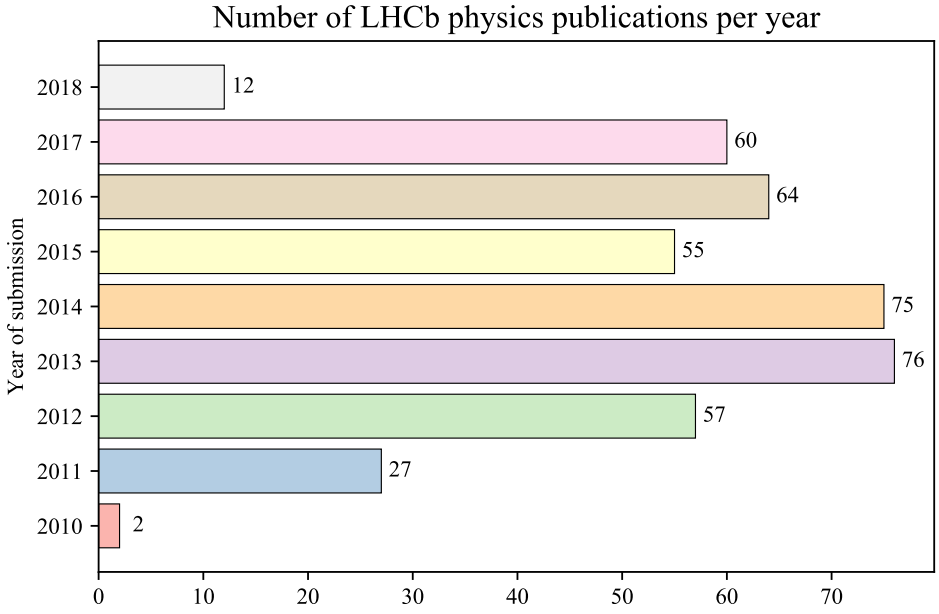


Figure 10.1: Number of physics papers published per year at LHCb. The figure was captured in May 2018.

Firstly, a solid infrastructure to capture active and completed analyses at LHCb is still needed. Currently, this information is captured in the working group databases, which exists in several different copies in different locations, and in the TWIKI portal. The current system is however not reliable, partly because there is no standardised way to document analyses. This results in a wide range of different TWIKI pages with no overall common structure. A new infrastructure should capture the pillars of preservation described in Chapter 8, which are basic information, data provenance, software repository and documentation (internal and public publications). It would provide a unique location linking to all analysis resources. Such

infrastructure can be implemented as a database, and it would simplify the tasks of the working group conveners, editorial board and the analysts themselves. In addition, such a database could provide a complete source of information for the CAP framework.

Secondly, as the LHCb experiment releases its data open access, the information needed to understand the data should be provided and be readily available to the general public. Guidelines on how to use the LHCb software and data should be provided as tutorials and analysis examples. Furthermore, it would be beneficial to external scholars if a selection of published analyses could be provided to illustrate how physics analyses are performed at the technical level. This can be implemented via the CERN Open Data portal.

Finally, it would be useful to work actively to raise awareness about analysis preservation and present best practices at collaboration workshops and other similar events. The CERN community is currently divided about the topic of analysis preservation. Generally speaking, the most numerous group is the one that does not know about the analysis preservation efforts. There is also a number of people who believe that analysis preservation is something intangible that should be explored mainly in the long-term future. Only a small, emerging community understands that analysis preservation can vastly facilitate performance of ongoing analyses and help with collaborative work. The goal of this study would be to promote the effort and identify the incentives for the particle physics community to commit to analysis preservation. This study may be partly shaped by the increasing pressure on project leaders from funding bodies, who more commonly require a data management and preservation plan [129, 198]. Since discussions and awareness-raising are not alone sufficient to secure preservation, these requirements are crucial external force.

We are moving towards higher automatisation in physics analyses, and in an optimal future scenario, we will be able to run complex analyses with the click of a button. Such automated analyses could be used at the beginning of data taking periods, to promptly learn about the properties of the particle collisions. In addition, they would have a great potential in reproducibility, reuse, education and outreach. We are going towards more preserved, more automated and more open science and the combined efforts of the CERN community will hopefully make this change possible.

Bibliography

[1] A. Trisovic. *Thesis resources*. `https://github.com/atrisovic/thesis-resources`. 2018.

[2] A. Trisovic et al. "Recording the LHCb data and software dependencies". *Journal of Physics: Conference Series*. Vol. 898. 10. IOP Publishing. 2017, p. 102010.

[3] A. Trisovic. *LHCb Monte Carlo simulation on the Cloud*. `https://github.com/atrisovic/lhcb-production-demo`. 2017.

[4] A. Trisovic. *Event comparison*. `https://github.com/atrisovic/lhcb-compare-events`. 2017.

[5] A. Trisovic. *Meta data service for LHCb software Gaudi*. `https://github.com/atrisovic/lhcb-metadatasvc`. 2017.

[6] A. Trisovic. *Analysis case study: reproducing an LHCb physics analysis*. `https://github.com/atrisovic/analysis-case-study`. 2017.

[7] M. Nielsen. *Reinventing discovery: the new era of networked science*. Princeton University Press, 2012.

[8] G. Aad et al. "Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC". *Physics Letters B* 716.1 (2012), pp. 1–29.

[9] S. Dittmaier and M. Schumacher. "The Higgs boson in the standard model—from LEP to LHC: expectations, searches, and discovery of a candidate". *Progress in Particle and Nuclear Physics* 70 (2013), pp. 1–54.

[10] The ATLAS collaboration. *The ATLAS Experiment at the CERN Large Hadron Collider, 2008*.

[11] R. Adolphi et al. "The CMS experiment at the CERN LHC, CMS collaboration". *Journal of Instrumentation* (2008).

[12] CERN. *The search for the Higgs boson*. https://timeline.web.cern.ch/timelines/The-search-for-the-Higgs-boson.

[13] S. Chatrchyan et al. "Observation of a new boson at a mass of 125 GeV with the CMS experiment at the LHC". *Physics Letters B* 716.1 (2012), pp. 30–61.

[14] J. Vitek and T. Kalibera. "Repeatability, reproducibility, and rigor in systems research". *EMSOFT*. ACM. 2011, pp. 33–38.

[15] C. Collberg, T. Proebsting, and A. M. Warren. *Repeatability and benefaction in computer systems research*. Tech. rep. Technical report, Univ. of Arizona TR 14-04, 2015.

[16] R. D. Peng. "Reproducible research in computational science". *Science* 334.6060 (2011), pp. 1226–1227.

[17] K. Hirata et al. "Observation of a neutrino burst from the supernova SN1987A". *Phys. Rev. Lett.* 58 (14 Apr. 1987), pp. 1490–1493.

[18] B. P. Abbott et al. "Observation of gravitational waves from a binary black hole merger". *Physical review letters* 116.6 (2016), p. 061102.

[19] M. Baker. "Is there a reproducibility crisis? A Nature survey lifts the lid on how researchers view the'crisis rocking science and what they think will help". *Nature* 533.7604 (2016), pp. 452–455.

[20] V. Stodden. "What scientific idea is ready for retirement: Reproducibility". *Edge Foundation* (2014).

[21] *Reproducibility in Science: A Guide to enhancing reproducibility in scientific results and writing*. http://ropensci.github.io/reproducibility-guide/.

[22] Wikipedia - the Free Encyclopaedia. *Large Electron–Positron Collider*. https://en.wikipedia.org/wiki/Large_Electron\OT1\textendashPositron_Collider. 2018.

[23] P. La Rocca and F. Riggi. "The upgrade programme of the major experiments at the Large Hadron Collider". *Journal of Physics: Conference Series*. Vol. 515. 1. IOP Publishing. 2014, p. 012012.

[24] R. Aaij et al. "First Observation of CP Violation in the Decays of B_s^0 Mesons". *Physical review letters* 110.22 (2013), p. 221601.

[25] The LHCb Collaboration. "Observation of $J/\psi p$ Resonances Consistent with Pentaquark States in $\Lambda_b^0 \to J/\psi K^- p$ Decays". *Phys. Rev. Lett.* 115 (7 Aug. 2015), p. 072001.

[26] CERN. *CERN Data Centre passes the 200-petabyte milestone*. `https://home.cern/about/updates/2017/07/cern-data-centre-passes-200-petabyte-milestone`. 2018.

[27] *The Challenges of Big (Science) Data*. `https://indico.cern.ch/event/466934/contributions/2524828/attachments/1490181/2315978/BigDataChallenges-EPS-Venice-080717.pdf`. 2017.

[28] *Data Redundancy*. `https://www.techopedia.com/definition/18707/data-redundancy`.

[29] J. Albrecht et al. "A Roadmap for HEP Software and Computing R&D for the 2020s". *arXiv preprint arXiv:1712.06982* (2017).

[30] C. The LHCb Collaboration. *Upgrade Software and Computing*. Tech. rep. CERN-LHCC-2018-007. LHCB-TDR-017. Geneva: CERN, Mar. 2018.

[31] C. Collberg et al. "Measuring reproducibility in computer systems research". *Department of Computer Science, University of Arizona, Tech. Rep* 37 (2014).

[32] Z. Akopov et al. "Status report of the DPHEP Study Group: Towards a global effort for sustainable data preservation in high-energy physics". *arXiv preprint arXiv:1205.4667* (2012).

[33] L. Lederman. "International Committee on Future Accelerators". *Fermilab report, Jul 1979, 8* 6 (1980).

[34] D. S. Group et al. "Data Preservation in High Energy Physics". *arXiv preprint arXiv:0912.0255* (2009).

[35] *DPHEP "H2020 brain-storming"*. `https://indico.cern.ch/event/315340/`. 2014.

[36] *DOI*. `https://www.doi.org`. 2018.

[37] *Open Access Policy for CERN Publications*. Tech. rep. CERN-OPEN-2017-020. Geneva: CERN, Apr. 2017.

[38] The LHCb collaboration and others. *LHCb External Data Access Policy. CERN Open Data Portal*. 2013.

[39] CMS collaboration (2012). *CMS data preservation, re-use and open access policy. CERN Open Data Portal*. 2012.

[40] The ATLAS experiment. *ATLAS Data Access Policy. CERN Open Data Portal*. 2014.

[41] I. Hrynaszkiewicz et al. "Standardising and harmonising research data policy in scholarly publishing". *bioRxiv* (2017), p. 122929.

[42] National Science Foundation (NSF). *Dissemination and Sharing of Research Results*. `https://www.nsf.gov/bfa/dias/policy/dmp.jsp`.

[43] Science and Technology Facilities Council. *Scientific Data Policy*. `http://www.stfc.ac.uk/about-us/our-purpose-and-priorities/freedom-of-information/scientific-data-policy/`. 2017.

[44] J. Cowton et al. "Open data and data analysis preservation services for LHC experiments". *Journal of Physics: Conference Series*. Vol. 664. 3. IOP Publishing. 2015, p. 032030.

[45] A. Buckley and M. Whalley. "HepData reloaded: reinventing the HEP data archive". *arXiv preprint arXiv:1006.0517* (2010).

[46] A. Ustyuzhanin et al. "Everware toolkit. Supporting reproducible science and challenge-driven education". *arXiv preprint arXiv:1703.01200* (2017).

[47] D. Piparo et al. *SWAN: a Service for Interactive Analysis in the Cloud*. Tech. rep. CERN-OPEN-2016-005. Geneva: CERN, June 2016.

[48] J. Kuncar, L. Nielsen, and T. Simko. "Invenio v2. 0: A Pythonic Framework for Large-Scale Digital Libraries Tech. Rep". ATLAS-CONF-2016-033 CERN Geneva URL http://urn. fi/URN: NBN: fi-fe2014070432294. 2014.

[49] *CERN makes public first data of LHC experiments*. `http://cds.cern.ch/record/1998990`. Nov. 2014.

[50] A. Larkoski et al. "Exposing the QCD Splitting Function with CMS Open Data". *arXiv preprint arXiv:1704.05066* (2017).

[51] A. Tripathee et al. "Jet Substructure Studies with CMS Open Data". *arXiv preprint arXiv:1704.05842* (2017).

[52] A. Trišović. "Measuring the D0 lifetime at the LHCb Masterclass". *Nuclear and Particle Physics Proceedings* 273 (2016), pp. 1215–1220.

[53] ATLAS. *Kaggle Higgs boson challenge*. `https://www.kaggle.com/c/higgs-boson`.

[54] LHCb and Yandex. *Kaggle Flavours of Physics challenge*. `https://www.kaggle.com/c/flavours-of-physics`.

[55] E. Maguire, L. Heinrich, and G. Watt. "HEPData: a repository for high energy physics data". *Journal of Physics: Conference Series*. Vol. 898. 10. IOP Publishing. 2017, p. 102006.

[56] *Measurement of J/ψ polarization in pp collisions at $\sqrt{s} = 7$ TeV*. Data Collection. 2013.

[57] M. Ragan-Kelley et al. "The Jupyter/IPython architecture: a unified view of computational research, from interactive exploration to communication and publication." *AGU Fall Meeting Abstracts*. 2014.

[58] Swan website. *Swan image*. `http://swan.web.cern.ch/notebook-galleries`. 2017.

[59] A. A. Alves Jr et al. "The LHCb detector at the LHC". *Journal of instrumentation* 3.08 (2008), S08005.

[60] L. Evans and P. Bryant. "LHC machine". *Journal of Instrumentation* 3.08 (2008), S08001.

[61] CERN. *Radiofrequency cavities*. `https://home.cern/about/engineering/radiofrequency-cavities`. 2017.

[62] CERN. *Engineering*. `https://home.cern/about/engineering`. 2018.

[63] CERN. *Pulling together: Superconducting electromagnets*. `https://home.cern/about/engineering/pulling-together-superconducting-electromagnets`. 2018.

[64] K. Aamodt et al. "The ALICE experiment at the CERN LHC". *Journal of Instrumentation* 3.08 (2008), S08002.

[65] Merk, Marcel. *The LHCb Upgrade*. `http://lhcb-public.web.cern.ch/lhcb/Physics-Results/Docs/Beauty2011-Merk.pdf`. 2011.

[66] R. M. Barnett, H. Mühry, and H. R. Quinn. *The Charm of Strange Quarks: Mysteries and Revolutions of Particle Physics*. Springer Science & Business Media, 2013.

[67] C. P. Enz and R. Lewis. "On the phenomenological description of CP violation for K-mesons and its consequences". *Helvetica Physica Acta (Switzerland)* 38 (1965).

[68] J. Back et al. "First observation of CP violation in the decays of B^{o}s mesons". *Physical Review Letters* 110.22 (2013), Article–number.

[69] CERN. *The matter-antimatter asymmetry problem*. `https://home.cern/topics/antimatter/matter-antimatter-asymmetry-problem`.

[70] M. Thomson. *Modern particle physics*. Cambridge University Press, 2013.

[71] A. D. Sakharov. "Violation of CP invariance, C asymmetry, and baryon asymmetry of the universe". *Soviet Physics Uspekhi* 34.5 (1991), p. 392.

[72] A. Dziurda. "Full offline reconstruction in real time with the LHCb detector". *EPJ Web of Conferences*. Vol. 127. EDP Sciences. 2016, p. 00007.

[73] A. Leflat. "Upgrade of the LHCb Vertex Locator". *Journal of Instrumentation* 9.08 (2014), p. C08015.

[74] C. Jones. *Private communication.*

[75] The LHCb Collaboration and others. "LHCb detector performance". *International Journal of Modern Physics A* (2015).

[76] R. Aaij et al. "Selection and processing of calibration samples to measure the particle identification performance of the LHCb experiment in Run 2". *arXiv preprint arXiv:1803.00824* (2018).

[77] LHCb, Technical design report. *RICH technical design report*. Tech. rep. CERN-LHCC-2000-037, 2000.

[78] M. Adinolfi et al. "Performance of the LHCb RICH detector at the LHC". *The European Physical Journal C* 73.5 (2013), p. 2431.

[79] The LHCb Collaboration. "Triggering with the LHCb calorimeters". *Journal of Physics: Conference Series*. Vol. 160. 1. IOP Publishing. 2009, p. 012063.

[80] F. Archilli et al. "Performance of the muon identification at LHCb". *Journal of Instrumentation* 8.10 (2013), P10020.

[81] The LHCb experiment on Instagram. *Photo Credit*. https://www.instagram.com/p/BX7mHz1F4YG.

[82] R. Antunes-Nobrega et al. "LHCb trigger system: Technical design report". *CERN/LHCC* 31 (2003).

[83] LHCb. *LHCb average Mu in pp in 2017*. https://lbgroups.cern.ch/online/OperationsPlots/index_files/2017AvgMu.png.

[84] V. Gligorov. *Private communication.*

[85] R. Aaij et al. "The LHCb trigger and its performance in 2011". *Journal of Instrumentation* 8.04 (2013), P04022.

[86] R. Aaij et al. "Tesla: an application for real-time data analysis in High Energy Physics". *Computer Physics Communications* 208 (2016), pp. 35–42.

[87] S. Benson et al. "The LHCb turbo stream". *Journal of Physics: Conference Series*. Vol. 664. 8. IOP Publishing. 2015, p. 082004.

[88] G. Dujany and B. Storaci. "Real-time alignment and calibration of the LHCb Detector in Run II". *J. Phys.: Conf. Ser.* 664.LHCb-PROC-2015-011. CERN-LHCb-PROC-2015-011 (Apr. 2015), 082010. 8 p.

[89] H. M. Evans. "Measurements of $B \to \mu^+\mu^-$ decays using the LHCb experiment". PhD thesis. University of Cambridge, 2017.

[90] G. Papotti et al. *Experience with offset collisions in the LHC*. Tech. rep. 2011.

[91] R. Alemany-Fernandez, F. Follin, and R. Jacobsson. "The LHCb online luminosity control and monitoring" (2013).

[92] *The LHCb public page*. http://lhcb-public.web.cern.ch/lhcb-public/. 2018.

[93] *ATLAS experiment: Luminosity summary plots for 2018 pp data taking*. https://twiki.cern.ch/twiki/bin/view/AtlasPublic/LuminosityPublicResultsRun2. 2018.

[94] C. Eck et al. *LHC computing Grid: Technical Design Report. Version 1.06 (20 Jun 2005)*. Technical Design Report LCG. Geneva: CERN, 2005.

[95] G. Merino et al. "Transition to a new CPU benchmarking unit for the WLCG". *HEPIX Benchmarking WK* (2009).

[96] The LHCb Collaboration and others. “The LHCb DIRAC-based production and data management operations systems”. *Journal of Physics: Conference Series*. Vol. 368. 1. IOP Publishing. 2012, p. 012010.

[97] T. Head et al. “The silicon vertex locator for the LHCb upgrade”. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 765 (2014), pp. 244–246.

[98] I. Bediaga et al. *LHCb Trigger and Online Upgrade Technical Design Report*. Tech. rep. CERN-LHCC-2014-016, 2014.

[99] The LHCb Collaboration and others. *LHCb tracker upgrade technical design report*. Tech. rep. 2014.

[100] The LHCb Collaboration and others. *LHCb VELO upgrade technical design report*. Tech. rep. 2013.

[101] LHC commissioning. *Performance 2016*. `https://lhc-commissioning.web.cern.ch/lhc-commissioning/performance/2016-performance.htm`.

[102] LHC commissioning. *Performance 2017*. `https://lhc-commissioning.web.cern.ch/lhc-commissioning/performance/2017-performance.htm`.

[103] G. L. Presti et al. “CASTOR: A Distributed Storage Resource Facility for High Performance Data Processing at CERN.” *MSST*. Vol. 7. 2007, pp. 275–280.

[104] L. Mascetti et al. “Disk storage at CERN”. *Journal of Physics: Conference Series*. Vol. 664. 4. IOP Publishing. 2015, p. 042035.

[105] G. Cancio et al. “Experiences and challenges running CERN’s high capacity tape archive”. *Journal of Physics: Conference Series*. Vol. 664. 4. IOP Publishing. 2015, p. 042006.

[106] Markus Frank. *Online Raw Data Format, LHCb Technical Note*. Tech. rep. 2006.

[107] P. Canal, B. Bockelman, and R. Brun. “ROOT I/O: The fast and furious”. *Journal of Physics: Conference Series*. Vol. 331. 4. IOP Publishing. 2011, p. 042005.

[108] M. Cattaneo et al. “The new LHCb Event Data Model”. *LHCb-2001-142* (2001).

[109] G. Corti. *Overview of Monte Carlo simulation(s) in LHCb*. `https://lhcb-comp.web.cern.ch/lhcb-comp/Simulation/Tutorial/01.GCorti-MCinLHCb-20091013.pdf`. 2009.

[110] M. Clemencic, J. Palacios, and N. Gilardi. *LHCb conditions database*. Tech. rep. 2006.

[111] G. Barrand et al. "GAUDI - A software architecture and framework for building HEP data processing applications". *Computer Physics Communications* 140.1 (2001), pp. 45–55.

[112] G. Corti et al. "Software for the LHCb experiment". *IEEE transactions on nuclear science* 53.3 (2006), pp. 1323–1328.

[113] M. Clemencic et al. "Recent developments in the LHCb software framework Gaudi". *Journal of Physics: Conference Series* 219.4 (2010), p. 042006.

[114] L. Collaboration et al. "The BRUNEL project". *Web page http://lhcb-releasearea. web. cern. ch/LHCb-release-area/DOC/brunel* (2017).

[115] L. Collaboration et al. "The MOORE project". *Web page http://lhcb-releasearea. web. cern. ch/LHCb-release-area/DOC/moore* (2012).

[116] L. Collaboration et al. "The DAVINCI project". *Web page http://lhcb-releasearea. web. cern. ch/LHCb-release-area/DOC/davinci* (2017).

[117] L. Collaboration et al. "The GAUSS project". *Web page http://lhcb-releasearea. web. cern. ch/LHCb-release-area/DOC/gauss* (2015).

[118] L. Collaboration et al. "The BOOLE project". *Web page http://lhcb-releasearea. web. cern. ch/LHCb-release-area/DOC/boole* (2017).

[119] L. Collaboration et al. *The PANORAMIX Project*.

[120] J. Shiers et al. *CERN Services for Long Term Data Preservation*. Tech. rep. 2016.

[121] J. Blomer, P. Buncic, and T. Fuhrmann. "CernVM-FS: delivering scientific software to globally distributed computing resources". *Proceedings of the first international workshop on Network-aware data management*. ACM. 2011, pp. 49–56.

[122] J. Blomer et al. "Status and future perspectives of CernVM-FS". *Journal of Physics: Conference Series*. Vol. 396. 5. IOP Publishing. 2012, p. 052013.

[123] *LHCb WG databases*. `https://lhcb-wg.web.cern.ch/lhcb-WG/`.

[124] *LHCb TWiki*. https://twiki.cern.ch/twiki/bin/view/LHCb/. 2018.

[125] L. Marian et al. *CERN Document Server*. Tech. rep. 2016.

[126] LHCb Q&A. https://lhcbqa.web.cern.ch/lhcbqa. 2017.

[127] *Provenance*. https://en.wikipedia.org/wiki/Provenance. 2018.

[128] T. Pasquier et al. "If these data could talk." *Scientific data* 4 (2017), p. 170114.

[129] M. D. Wilkinson et al. "The FAIR Guiding Principles for scientific data management and stewardship". *Scientific Data* 3 (Mar. 2016), p. 160018.

[130] K. Belhajjame et al. *Prov-DM: The PROV Data Model*. Tech. rep. World Wide Web Consortium (W3C), 2013.

[131] R. A. Nobrega et al. "LHCb computing technical design report" (2005).

[132] T. Sjöstrand, S. Mrenna, and P. Skands. "PYTHIA 6.4 physics and manual". *Journal of High Energy Physics* 2006.05 (2006), p. 026.

[133] D. J. Lange. "The EvtGen particle decay simulation package". *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 462.1 (2001), pp. 152–155.

[134] S. Agostinelli et al. "GEANT4—a simulation toolkit". *Nuclear instruments and methods in physics research section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 506.3 (2003), pp. 250–303.

[135] J. Allison et al. "Geant4 developments and applications". *IEEE Transactions on nuclear science* 53.1 (2006), pp. 270–278.

[136] J. Webber. "A programmatic introduction to Neo4j". *Proceedings of the 3rd annual conference on Systems, programming, and applications: software for humanity*. ACM. 2012, pp. 217–218.

[137] *JSON*. https://en.wikipedia.org/wiki/JSON. 2018.

[138] M. Clemencic and B. Couturier. "Deployment Infrastructure for run 2". *Journal of Physics: Conference Series*. Vol. 664. 2015, p. 062008.

[139] M. Grinberg. *Flask web development: developing web applications with python*. " O'Reilly Media, Inc.", 2018.

[140] N. Small. *Py2neo*. 2016.

[141] M. Jakl. “Representational State Transfer (REST)”. *Pro PHP XML and Web Services SE-17* (2006), p. 633.

[142] S. Dallmeier Tiessen et al. *CERN analysis preservation (CAP) – Use Cases*. 2015.

[143] P. Mell, T. Grance, et al. “The NIST definition of cloud computing” (2011).

[144] The Economist. *The cheap, convenient cloud*. `https://www.economist.com/business/2015/04/18/the-cheap-convenient-cloud`. 2015.

[145] O. Sefraoui, M. Aissaoui, and M. Eleuldj. “OpenStack: toward an open-source solution for cloud computing”. *International Journal of Computer Applications* 55.3 (2012).

[146] D. Bernstein. “Containers and cloud: From LXC to Docker to Kubernetes”. *IEEE Cloud Computing* 1.3 (2014), pp. 81–84.

[147] J. Blomer et al. “Micro-CernVM: slashing the cost of building and deploying virtual machines”. *Journal of Physics: Conference Series*. Vol. 513. 3. IOP Publishing. 2014, p. 032009.

[148] Docker Inc. *The official web page of the Docker technology*. `https://www.docker.com/`. 2017.

[149] *DockerHub registry*. `https://hub.docker.com`. 2018.

[150] E. A. Brewer. “Kubernetes and the path to cloud native”. *Proceedings of the Sixth ACM Symposium on Cloud Computing*. ACM. 2015, pp. 167–167.

[151] REANA. *Reusable analysis example – ROOT6 and RooFit*. `https://github.com/reanahub/reana-demo-root6-roofit`. 2017.

[152] R. Brun and F. Rademakers. “ROOT—an object oriented data analysis framework”. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 389.1-2 (1997), pp. 81–86.

[153] R. Brun et al. *ROOT users guide 5.16*. 2007.

[154] M. Cattaneo et al. “Gaudi Developers Guide”. *December* 20.2 (2001), o01.

[155] A. Trisovic. *Implementation of the MetaData service in the Gaudi framework*. `https://gitlab.cern.ch/lhcb/Gaudi/tree/future/GaudiSvc/src/MetaDataSvc` and `https://gitlab.cern.ch/lhcb/Gaudi/blob/future/GaudiSvc/doc/MetaDataSvc.md`. 2015.

[156] *Hypertext Markup Language (HTML)*. https://en.wikipedia.org/wiki/HTML. 2018.

[157] The LHCb collaboration. "Search for $D^+_{(s)} \to \pi^+\mu^+\mu^-$ and $D^+_{(s)} \to \pi^-\mu^+\mu^+$ decays". *Phys. Lett. B* 724 (Apr. 2013), 203–212. 21 p.

[158] E. Greening. "Rare and Challenging Charm Decays at LHCb". Presented 27 Aug 2014. PhD thesis. University of Oxford, July 2014.

[159] E. Greening et al. "Searches for $D^\pm_{(s)} \to h^\pm\mu^+\mu^-$ and $D^\pm_{(s)} \to h^\pm\mu^\pm\mu^\pm$" (Apr. 2013). Linked to LHCb-PAPER-2012-051.

[160] V. Abazov et al. "Search for flavor-changing-neutral-current D meson decays". *Physical review letters* 100.10 (2008), p. 101801.

[161] J. Link et al. "Search for rare and forbidden 3-body di-muon decays of the charmed mesons D^+ and D_s^+". *Physics Letters B* 572.1-2 (2003), pp. 21–31.

[162] *Official LHCb stripping website*. http://lhcb-release-area.web.cern.ch/LHCb-release-area/DOC/stripping/config/stripping21r1/charm/strippingd2xmumu_piosline.html. 2018.

[163] A. Hoecker et al. "TMVA – Toolkit for multivariate data analysis". *arXiv preprint physics/0703039* (2007).

[164] A. Paul, I. I. Bigi, and S. Recksiegel. "On $D \to X_u l^+ l^-$ within the Standard Model and frameworks like the littlest Higgs model with T parity". *Physical Review D* 83.11 (2011), p. 114006.

[165] *Invariant mass*. https://en.wikipedia.org/wiki/Invariant_mass. 2018.

[166] E. Greening. *Rare Charm Decays at LHCb*. https://cds.cern.ch/record/1596731. Sept. 2013.

[167] M. Baker. "1,500 scientists lift the lid on reproducibility". *Nature* 533.7604 (2016), pp. 452–454.

[168] J. Kovacevic. "How to encourage and publish reproducible research". *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*. Vol. 4. IEEE. 2007, pp. IV–1273.

[169] C. Collberg and T. A. Proebsting. "Repeatability in computer systems research". *Commun ACM* 59.3 (2016), pp. 62–69.

[170] *GitHub*. `https://github.com`. 2018.

[171] *Bitbucket*. `https://bitbucket.org`. 2018.

[172] Wikipedia - the Free Encyclopaedia. *README file*. `https://en.wikipedia.org/wiki/README`. 2018.

[173] *Read the Docs. Create, host, and browse documentation.* `https://readthedocs.org`. 2018.

[174] *The ROOT Framework to be released in Docker*. `https://root.cern.ch/root-docker-container-alpha-version`. 2018.

[175] P. Buncic et al. "CernVM–a virtual software appliance for LHC applications". *Journal of Physics: Conference Series*. Vol. 219. 4. IOP Publishing. 2010, p. 042003.

[176] G. M. Kurtzer, V. Sochat, and M. W. Bauer. "Singularity: Scientific containers for mobility of compute". *PloS one* 12.5 (2017), e0177459.

[177] J. Köster and S. Rahmann. "Snakemake—a scalable bioinformatics workflow engine". *Bioinformatics* 28.19 (2012), pp. 2520–2522.

[178] P. Amstutz et al. "Common Workflow Language, v1. 0" (2016).

[179] K. Cranmer and L. Heinrich. "Yadage and Packtivity–analysis preservation using parametrized workflows". *Journal of Physics: Conference Series*. Vol. 898. 10. IOP Publishing. 2017, p. 102019.

[180] *REANA demo example - BSM search*. `https://github.com/reanahub/reana-demo-bsm-search`. 2018.

[181] Wikipedia - the Free Encyclopaedia. *YAML*. `https://en.wikipedia.org/wiki/YAML`. 2018.

[182] *CERN Mattermost*. `https://mattermost.web.cern.ch`. 2018.

[183] *JIRA*. `https://sft.its.cern.ch/jira`. 2018.

[184] A. Puig and L. S. Team. "The LHCb Starterkit". *Journal of Physics: Conference Series*. Vol. 898. 8. IOP Publishing. 2017, p. 082054.

[185] CERN Analysis Preservation: CAP Client. *command-line client used for communication with the CERN Analysis Preservation server*. https://github.com/cernanalysispreservation/cap-client. 2017.

[186] *The REANA project*. http://www.reana.io. 2018.

[187] A. Trisovic. *Git integration*. https://github.com/cernanalysispreservation/analysispreservation.cern.ch/issues/589. 2018.

[188] *Python GitLab documentation*. https://python-gitlab.readthedocs.io/en/stable/. 2018.

[189] *PyGitHub documentation*. http://pygithub.readthedocs.io/en/latest/index.html. 2018.

[190] *Skopeo*. https://github.com/projectatomic/skopeo. 2018.

[191] A. Trisovic. *Survey on analysis reproducibility: data and code*. https://github.com/atrisovic/thesis-plots. 2018.

[192] Wikipedia - the Free Encyclopaedia. *Makefile*. https://en.wikipedia.org/wiki/Makefile. 2018.

[193] *Snakemake documentation*. https://snakemake.readthedocs.io/en/v5.1.4/. 2018.

[194] Wikipedia - the Free Encyclopaedia. *R programming language*. https://en.wikipedia.org/wiki/R_(programming_language). 2018.

[195] *Computing with HTCondor*. https://research.cs.wisc.edu/htcondor/index.html. 2018.

[196] S. Binet and B. Couturier. "docker & HEP: Containerization of applications for development, distribution and preservation". *Journal of Physics: Conference Series*. Vol. 664. 2. IOP Publishing. 2015, p. 022007.

[197] S. Neubert et al. *LHCb Analysis Preservation Roadmap*. Tech. rep. LHCb-INT-2017-021. CERN-LHCb-INT-2017-021. Geneva: CERN, Aug. 2017.

[198] Q. Schiermeier. "Data management made simple". 555 (Mar. 2018), pp. 403–405.