Python2 to 3: migration and the improvement of the WMAgentScripts repository for CMS Computing Operations

Author:
Amanda Goldani R. Peixoto

Supervisors:
Hasan Öztürk
Zhangqier Wang
Jennifer Adelman-McCarthy

CERN Summer Programme
August 2021

1 Overview

The Compact Muon Solenoid (CMS) experiment consists in a multi-purpose detector located at the Large Hadron Collider (LHC), at CERN. It collects data from collisions occurring every 25ns during an LHC run. However, even in a long shutdown period, a huge amount of events is still processed and analyzed by the CMS experiment. It reaches around 2PB per month — if considered collision data and Monte Carlo simulations. Hence, the CMS production is continuously running hundreds of thousands of jobs in the Worldwide LHC Computing Grid (WLCG) [1], using up to 300k of the available processor cores.

The production and reprocessing of data workflows through the computing infrastructure include several steps: from the transfer of input datasets to the sites where they are needed at, until the delivery of output datasets to CMS Physics groups around the globe [2]. A workflow can be described as the state machine in Figure 1. It also belongs to a complex computing framework, which relies on several services maintained by different teams at CERN.

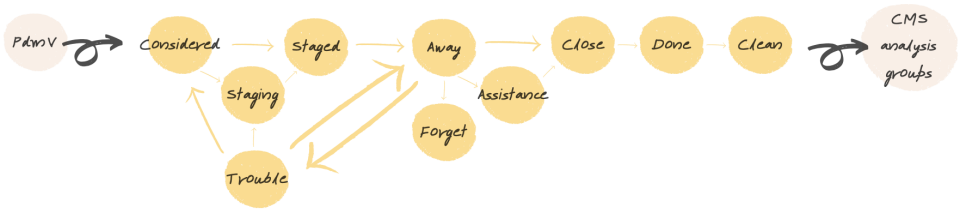

Figure 1: Diagram of the state machine describing a workflow through the computing infrastructure. Workflow requests are approved by the Physics Data Monte Carlo Validation (PdmV) Team; output datasets are distributed to CMS analysis groups. Adapted from [2].

Amongst those, the Production & Reprocessing (P&R) Team is responsible for operating the central production tools towards the delivery of data to the CMS analysis groups. It also develops and maintains the scripts of an important production software, which centralizes some key operational procedures. These include, for example, the assignment and the completion announcement of a given workflow.

The mentioned scripts are stored in the WMAgentScripts repository [1] in GitHub and were entirely written in Python2 a few years ago. The repository contains more than 46k lines of code, with 45 modules in its core functionality relentlessly running through 5 production agents.

[1]The repository link and other related links are provided at the end of the report.

2 Motivation & Objectives

As of January 1st, 2020, Python2 stopped being officially supported by the Python Software Foundation [3]. Considering that Python3 is not backwards-compatible, all the Python community is since then moving to Python3 and third-party libraries have stopped providing support to Python2 as well. By consequence, the development and maintenance of Python2 code tend to become harder with time, while incompatibility and vulnerability issues may also increase [4].

In this context, the switch to Python3 became a necessity to most organizations still running Python2 code. Although resource and time demanding, the upgrade offers a good opportunity to explore the better performance of Python3 and its new features. It may also provide a convenient moment for the review and modernization of existing code.

As such, the objectives of this project were to begin the migration of the WMAgentScripts repository from Python2 to Python3. In particular, it should focus on the migration of the helper functions and classes used by the repository's core functionalities. These were originally stored in a single file summing up more than 5k lines, so that the mapping of weak parts as well as the proposal of improvements were also targeted.

3 Approach to the Python3 Migration

It is important to note that none of the available automatic migration tools — such as 2to3 or six — were used, as the code improvement was also of interest during the version migration and there was no need for the software to support both versions. Instead, a guide [5] and the Python documentation [6] were used as guideline for understanding the differences between Python2 and Python3.

Also, coding best practices were always kept in mind. Mainly as a way to improve the code maintenance and development during migration but especially in the long term. Amongst these practices, one can highlight the use of a well defined code style, where the official Python proposals were mostly followed [7]; and the use of type hints and annotations [8], which were only introduced in Python3 and can increase static code readability and documentation while requiring small development efforts. Clean and reusable code principles were also extensively considered [9, 10].

Additionally, some quality assurance tools were used to support the migration process. These included the use of an automatic code formatter [11] to ensure code style consistency; a specific Git development branch; and testing. A particular regard was given to this latter point. This because, besides the WMAgentScripts repository being a mature software, it does not include a proper testing environment yet. Consequently, unit tests needed to be

written from scratch in order to prevent fully manual verification — which can be very error prone.

4 Completed Work

Progress of Python3 Migration

All the helper functions and classes contained in the previously mentioned repository's file have already been migrated to Python3, as well as the functions from other smaller files where there existed direct dependency. This corresponds to around 13% of the WMAgentScripts repository, in terms of number of lines.

Unit Testing

For the code migration, unit tests were written based on the original Python2 behavior when permanent data was available and/or the functionality was related to a get method. As a result, the Python3 version of the code already counts with unit tests, with a total coverage of 54%. This provides more confidence to the migration process, since having "tests eliminates the fear that cleaning up the code will break it" [9].

Code Improvement

Python's design principles include the aphorisms "Flat is better than nested. Sparse is better than dense" [12]. In this context, the dense Python2 file containing almost all the helper functions of the repository's core functionality should be restructured. The original functions were then split into packages and modules, and they were grouped together based on their purpose. Although still a work in progress, this new code structure (see Figure 2) provides the benefit of letting the code more sustainable and of improving its overall maintainability.

Figure 2: Diagram of the change in the code structure during the migration process.

Also, readability is an important concern when writing cleaner code [9]. In this sense, other code improvements made during this migration step included renaming functions and classes when necessary, dropping deprecated functionalities, and writing simpler functions when possible.

5 Conclusion & Future Work

There is still a lot of work to be done before the whole Python3 version of the WMAgentScripts repository can be deployed to production. Since the helper functions used by the repository's core functionality were already migrated to Python3, the next step should be to start the migration of one of the repository's core modules.

Also, regarding what has already been migrated, some points still have scope for improvements and need to be further discussed in the future. Therefore, the related issues were already mapped in GitHub.

Finally, it is essential to enhance the testing in the WMAgentScripts repository. The unit tests already written to support the Python3 migration are just a start point. It is important to grow their coverage but also to implement integration tests and develop a proper test-bed environment, so that the code development can be done more confidently in the future.

6 Acknowledgments

To dedicate the summer working on this project was a great experience and this would not be possible without the support of some people. Firstly, I would like to thank very much my supervisors Zhangqier Wang, Jennifer Adelman-McCarthy, and especially Hasan Öztürk — for kindly answering all my questions and helping me to improve my coding skills. I cannot thank you enough for allowing me to have this unique opportunity.

Also, I am thankful to the CERN Summer Programme Team — for all their hard work in making this possible —, and to the CERN & Society donors — for largely supporting the Non-Member States studentships.

I certainly look forward to continue working on this project and to embrace new challenges in the future.

Related links

The WMAgentScripts repository is maintained in GitHub by the CMS Computing Operations and can be reached by this link. My work throughout the summer resulted in five pull requests, which can be checked in the following links: PR1, PR2, PR3, PR4, PR5. Finally, to provide support for the continuation of this project, detailed migration notes can be found in a GitHub wiki page by this link.

References

[1] Computing Grid. *CMS Experiment*, n. d. Accessed on Aug 2021, available at: https://cms.cern/detector/computing-grid.

[2] VILMANT, Jean-Roch et al. Software and experience with managing workflows for the computing operation of the CMS experiment. *Journal of Physics: Conference Series*. IOP Publishing, 2017. p. 052025.

[3] Sunsetting Python 2. *Python. org*, n. d. Accessed on Aug 2021, available at: https://www.python.org/doc/sunset-python-2/.

[4] Time to shed Python2. *National Cyber Security Center*, 2019. Accessed on Aug 2021, available at: https://www.ncsc.gov.uk/blog-post/time-to-shed-python-2.

[5] REGEBRO, Lennart. *Supporting Python 3: an in-depth guide*. Colliberty, 2020. Accessed on Aug 2021, available at: http://python3porting.com.

[6] What's new in Python. *Python. org*, n. d. Accessed on Aug 2021, available at: https://docs.python.org/3/whatsnew/index.html#whatsnew-index.

[7] VAN ROSSUM, Guido; WARSAW, Barry; COGHLAN, Nick. PEP 8: style guide for Python code. *Python. org*, 2001. Available at: https://www.python.org/dev/peps/pep-0008/.

[8] VAN ROSSUM, Guido; LEHTOSALO, Jukka; Langa, Łukasz. PEP 484: type hints. *Python. org*, 2014. Available at: https://www.python.org/dev/peps/pep-0484/.

[9] MARTIN, Robert C. *Clean code: a handbook of agile software craftsmanship*. Pearson Education, 2009.

[10] GAMMA, Erich et al. *Elements of reusable object-oriented software*. Reading, Massachusetts: Addison-Wesley, 1995.

[11] The uncompromising code formatter. *Black*, n. d. Accessed on Aug 2021, available at: https://black.readthedocs.io/en/stable/.

[12] PETERS, Tim. PEP 20: the zen of Python. *Python. org*, 2004. Accessed on Aug 2021, available at: https://www.python.org/dev/peps/pep-0020/.