Prompt calibration automation at Belle II

David Dossett[1,*] and *Martin* Sevior[1,†] for the Belle II Collaboration

[1]School of Physics (David Caro Building), The University of Melbourne, VIC 3010, Australia

Abstract. The Belle II detector began collecting data from e^+e^- collisions at the SuperKEKB electron-positron collider in March 2019. Belle II aims to collect a data sample 50 times larger than the previous generation of B-factories. For Belle II analyses to be competitive it is crucial that calibration payloads for this data are calculated promptly prior to data reconstruction. To accomplish this goal a Python plugin package has been developed based on the open-source Apache Airflow package; using Directed Acyclic Graphs (DAGs) to describe the ordering of processes and Flask to provide administration and job submission web pages. DAGs for calibration process submission, monitoring of incoming data files, and validation of calibration payloads have all been created to help automate the calibration procedure. Flask plugin classes have been developed to extend the built-in Airflow administration and monitoring web pages. Authentication was included through the use of the pre-existing X.509 grid certificates of Belle II users.

1 Introduction

In March 2019 the Belle II detector began collecting data from e^+e^- collisions at the SuperKEKB electron-positron collider [1]. The Belle II experiment uses the asymmetric (4 GeV e^+, 7 GeV e^-) SuperKEKB collider at the High Energy Accelerator Research Organization (KEK) in Tsukuba, Japan. The planned peak luminosity is $8 \times 10^{35}\,\mathrm{cm}^{-2}\,\mathrm{s}^{-1}$, with a final integrated luminosity of $50\,\mathrm{ab}^{-1}$ [2].

At Belle II excellent calibration of the data prior to analysis is crucial to keep the results competitive with LHCb [3]. Particle Identification (PID) efficiency and B meson vertex resolution strongly affect the statistical power of most physics analyses. Therefore it is necessary to have the best calibrated data reprocessed several times throughout the year. However there are benefits to providing prompt datasets, available on a timescale of approximately two weeks, that are as close to publication quality as possible. Firstly it allows physicists to begin their analyses early and then switch to official publication datasets for the final results. Secondly, the calibrated prompt datasets also provide an opportunity for early performance measurement of the Belle II detector so that any problems can be more quickly identified.

The Belle II Analysis Software Framework (basf2) [4] is a C++ framework that provides common tools for manipulating Belle II data. A Python API is also provided so that end-users can define the chain of C++ calculations to be run on each event easily. In order to provide a common framework for calibration experts, the Calibration and Alignment Framework (CAF)

[*]e-mail: david.dosssett@unimelb.edu.au
[†]e-mail: martines@unimelb.edu.au

was created in basf2 [5]. It provides an API to easily parallelize job submission of CAF basf2 processes on different batch system backends and is used by most calibrations at Belle II. It allows processing of large amounts of data to create calibration payload files, similar to the workflows used for automated calibration at the CMS and LHCb experiments [6, 7], but the CAF can be run at any computing centre easily so long as the necessary input data is located there.

2 Prompt calibration

2.1 Belle II data processing

At Belle II the data taking periods are defined by *experiments* and *runs*, which are given incrementing integer numbers *e.g.* (experiment=10, run=3003). An experiment is usually defined as a longer period of O(month) with consistent detector conditions. It is usually incremented when a major detector/accelerator change occurred, or after a shutdown. The run number is incremented on demand from the Belle II Data Acquisition (DAQ) system and corresponds to $\leq$ 8 hours of data taking.

The conditions database for Belle II allows calibration constants to be retrieved using a name, called a global tag [8]. Calibration payloads are then assigned to experiment and run number ranges called Intervals of Validity (IoVs). The combination of global tag, experiment number, and run number is used to identify the calibration constants to be used during a basf2 process. It is the goal of any calibration procedure to define a set of payloads for an overall range of run numbers.

Track and cluster reconstruction is performed only once during the data production chain; when converting from raw data to a data file format called *mDST*. Track and cluster objects are stored in the mDST file format, while the raw data objects used to derive them are removed. These reconstructed objects are re-used when during both the event reduction processing (analysis skimming) and user level analysis to produce N-tuples. Therefore all calibration constants for a data taking period (a range of run numbers) that affect reconstruction must be calculated prior to the relevant mDST production.

2.2 Current prompt calibration procedure

For real collision data there are two types of datasets produced for Belle II, prompt and reprocessed. Reprocessed data is acceptable for use in publication but is only produced in large amounts once or twice per year. Prompt datasets are produced more continuously, by calculating a subset of payloads for new runs and then performing the reconstruction to produce mDST files, see Fig. 1.

The calibration payloads that are derived in the prompt calibration are mostly the ones that can run successfully on data collected in a preceding two week period. Each one has different event type and integrated luminosity requirements leading to a set of data sources that must be usable in a single prompt calibration. For example the Vertex Detector (VXD) alignment requires a mixture of collision and cosmic events, while some other calibrations use events flagged by the High Level Trigger (HLT) as $e^+e^- \rightarrow e^+e^-$ (Bhabha), $e^+e^- \rightarrow \mu^+\mu^-$, or $e^+e^- \rightarrow \gamma\gamma$.

Prior to prompt processing several calibration *skims* of raw data are made by using the HLT flags mentioned above to filter calibration event types into separate files. This is a very fast operation as no reconstruction is performed, only previously calculated HLT flags are used to write out copies of matching events. The skimmed files are produced automatically and kept for 6 months before being discarded.

The prompt calibration process has several overall steps:

1. Define the run range of the prompt processing and a temporary *staging* global tag to store the new constants.

2. Upload any new local calibration payloads produced by detector experts. These are derived from data taken locally by individual detectors without using the DAQ.

3. Calibrations that affect the reconstruction of tracks and clusters the most are performed *e.g.* alignment.

4. Production of new files in a custom file format, *cDST*, using data from the prompt run range. These files contain reconstructed tracks and other common objects useful to later calibrations.

5. The calibrations that can use cDST files as input are run. Since they use the reconstructed objects derived in step 4, which are approximately calibrated in step 3, they don't need to perform the reconstruction again and use fewer computing resources.

Within the calibration steps 3 and 5 above there may also be dependencies between calibrations where payloads must be uploaded to the conditions database and used as input to dependent calibration processes.

Many different experts are required to run the calibration processes properly. The experts are assigned issue tracker tickets in the Jira system. They describe the IoV of the prompt processing, the location of data files, the staging global tag and many other necessary details so that the constants will be correctly determined. A prompt processing manager is responsible for the assigning of tickets and initial details. Detector experts are responsible for validating that the payloads give good corrections, and that there are no gaps in the IoV of the payloads uploaded to the conditions database.

3 Automating The prompt calibration

3.1 Problems facing the current situation

Experience has shown that the above procedure is often delayed by human error. If an expert accidentally used the wrong global tag or data files in a script, it was possible for incorrect payloads to be uploaded to the conditions database. Once the error was noticed it was difficult

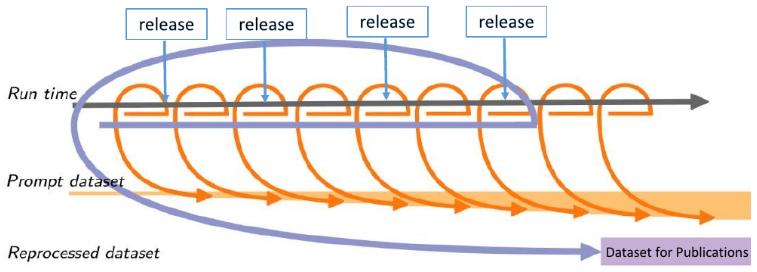


Figure 1. The rough scheme of prompt processing vs. major reprocessing. Releases are basf2 software versions, usually minor or patch release versions, which are requested whenever calibration code is updated. Therefore the total prompt dataset will contain data processed using several basf2 versions. A new reprocessed dataset is made every ~ 6 months using old + new data and the best calibration constants available.

to reset the staging global tag back to the correct state and restart from the correct point in the workflow. Other errors come from having to interact manually with the conditions database. If a calibration expert fails to create payloads with IoVs that cover the full set of runs defined by the prompt processing, then it is possible for some mDST production jobs to fail due to missing payloads.

Additionally, there was no database of calibration input data that allowed querying of files based on metadata about the data conditions *e.g.* run quality or run type. Due to this, experts have found it difficult to create lists of the correct data files for their calibration and to create scripts to use them. This introduced frequent errors where incorrect input data was used for a calibration, which either delayed a calibration's results or caused a restart as above.

CAF scripts have also often been left uncommitted by calibration experts. This necessarily leads to a lack of bookkeeping about which scripts were used for each calibration. It also means that it cannot be easily checked if the correct basf2 release was used, which may cause compatibility issues between the release used for mDST production and the payloads that were uploaded. As part of the push for automation, experts were asked to provide prompt CAF scripts to be included in basf2 releases. Any calibration without a script would not be run by the final automated system.

Jira tickets have been successful for managing and monitoring the overall prompt procedure. Calibration experts use Jira for various development tasks already, and it is an excellent tool for having public discussions of issues. However they do not strictly enforce that the processes will be run correctly by the experts. It was requested that automation should reduce the manual parts of running the prompt calibration processes, but keep the Jira ticketing system for monitoring.

3.2 Airflow overview

After comparing several commercial workflow tools, Apache Airflow [9] was chosen as the tool for the automation. Airflow is an open source Python package created by the company Airbnb. It provides an expressive way to define processing workflows (pipelines) as Directed Acyclic Graphs (DAGs) in pure Python 3, these workflows are themselves referred to as DAGs later in the text. Using Python immediately allows the use of many prebuilt packages for interfacing with tools such as LDAP and Jira. Airflow was chosen over several competing tools primarily due to its built-in features, such as:

- A built-in scheduler so that DAGs can be run on a regular cron-like schedule, or triggered when required.
- A Flask based administration web server including a Create-Read-Update-Delete (CRUD) interface to the database and DAG monitoring.
- Many classes supporting different process types *e.g.* Python, Bash, SSH.
- Easy plugin extension of both the Flask web server and processing classes.
- Ability to scale concurrent tasks by using the Celery task queue [10] as a process executor.

In Airflow's model a DAG is defined in Python and it contains one or more instances of an *operator* class. These operators may depend on each other and can be set to branch, wait for success/failure of parent operators, or reschedule themselves while waiting for a specific condition to occur (also known as a *sensor*).

3.3 Prompt calibration in Airlfow

In order to automate the procedure defined in Section 2.2 a Python package called *b2cal* has been created. This includes a calibration MySQL database defined using SQLAlchemy classes [11], DAGs, web pages, and some common basf2 scripts. The database contains tables for basic run conditions, data directories, data metadata, prompt processing requests, calibrations with assigned experts, associated Jira tasks *etc.* DAGs perform the asynchronous processing, including:

- Collecting run information from the Belle II run registry REST API, as well as discovering data file paths at the KEK Computing Centre (KEKCC).
- Collecting calibration scripts from new basf2 releases installed on the CernVM File System (CVMFS) [12].
- Submitting calibration and cDST production jobs remotely at KEKCC.
- Checking calibration payload IoVs for gaps before allowing upload to the conditions database.
- Initial Jira ticket creation, for all calibration and cDST productions that are requested to be in the prompt calibration.
- Uploading the signed off payloads to the conditions database, see Fig. 2.

Airflow's Flask web server has been extended with many custom web pages for defining and interacting with a prompt processing request. All of the Jira tickets are automatically created after an initial prompt processing request is made, with blocking links from and to each Jira task also added. However progress on a Jira ticket is only started, and the expert assigned to it, when a the relevant task in b2cal has no unfinished dependencies left. Once a task has started in b2cal the expert can then visit web pages to submit jobs. The experts are presented with web forms allowing them to search for input data based on some common run conditions, but also based on a basic metadata system (*data tags*), see Fig. 3. Some tags are assigned automatically to data directories when they are inserted to the database *e.g.* HLT skim names. It is also valid for an expert to request that a set of paths receive a tag for blacklist or whitelist purposes.

After CAF or data production jobs are completed, the experts are notified on the Jira tickets and the results can be downloaded or viewed from the b2cal web pages. Once the expert assigned to a task is happy that the results are consistent they can sign off, allowing any downstream tasks to begin, see Fig. 2.

3.4 Early operation

A first test of this full automated prompt calibration has been made using the winter 2019 data, comprising of $\sim 4\,\mathrm{fb}^{-1}$. The feedback from the calibration experts was very positive with some requests for additional features *e.g.* for more information on output logging to be notified on Jira tickets.

In order to allow secure access to the development website hosted at Melbourne, grid certificate (X.509) access was implemented using Airflow's plugin system. User details from the DIRAC [13] system used by Belle II's distributed computing are collected, inserted into the MySQL database, and used to uniquely identify an incoming grid certificate. This prevents users from needing to create new accounts, and means that no secure user information is stored on the server. In the future, the web server will be moved to the Deutsches Elektronen-Synchrotron laboratory (DESY) where most of the Belle II collaborative services are hosted. DESY provides an internal LDAP server for use by all collaborative services, and the web server login will be converted to use this instead of grid certificates.

Calibration experts must validate that the output payloads are satisfactory. This is usually achieved either by monitoring plots created during the CAF process, or by running test basf2

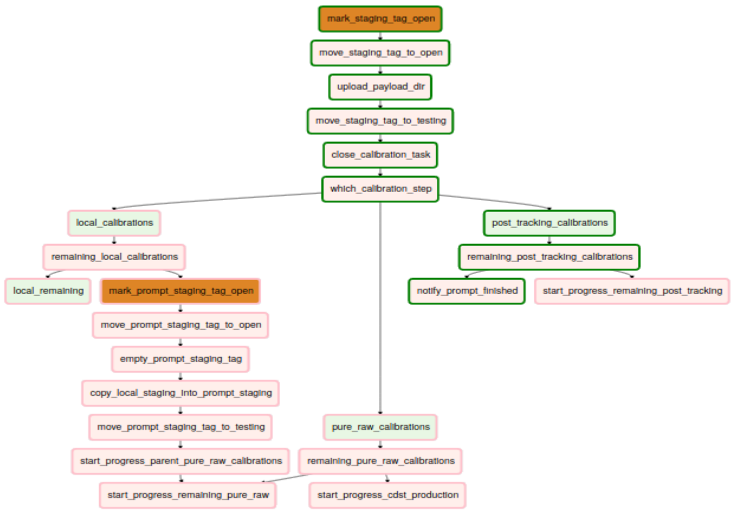
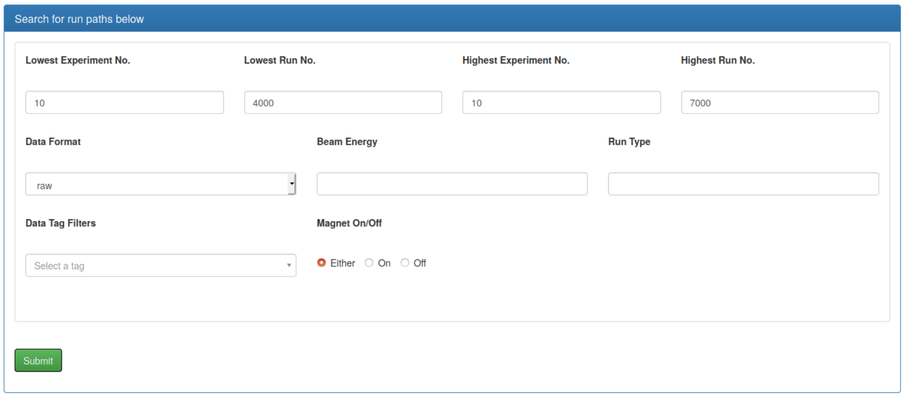

mark_staging_tag_open
move_staging_tag_to_open
upload_payload_dir
move_staging_tag_to_testing
close_calibration_task
which_calibration_step
local_calibrations
remaining_local_calibrations
local_remaining
mark_prompt_staging_tag_open
move_prompt_staging_tag_to_open
empty_prompt_staging_tag
copy_local_staging_into_prompt_staging
move_prompt_staging_tag_to_testing
start_progress_parent_pure_raw_calibrations
start_progress_remaining_pure_raw
post_tracking_calibrations
remaining_post_tracking_calibrations
notify_prompt_finished
start_progress_remaining_post_tracking
pure_raw_calibrations
remaining_pure_raw_calibrations
start_progress_cdst_production

Figure 2. Part of the DAG monitoring page showing the operators that run when a calibration was completed. A green border is a completed process, pink is a skipped process, orange processes are sensors and will not complete until they return a specific value. In this case it was the final calibration of the prompt processing request, so the payloads of the last completed calibration were uploaded and the completion of the prompt processing was notified on a Jira ’Epic’ task.


Search for run paths below

Lowest Experiment No.	Lowest Run No.	Highest Experiment No.	Highest Run No.
10	4000	10	7000

Data Format	Beam Energy	Run Type
raw		

Data Tag Filters: Select a tag

Magnet On/Off: [✓ Either] On Off

Figure 3. An example web form used to search for calibration input data. Multiple options and data tags can be selected to query for exactly the data required. For calibrations that require multiple categories of input data a web form like this is generated for each one. This allows for independent filtering of input data for each category.

processes on data to produce monitoring plots. We will be investigating how to allow experts to define and download the expected CAF plots from a submitted job. As well as automatically running a calibration-dependent basf2 process after a successful CAF job has created new payloads. The validation tasks and scripts are expected to become better defined by the relevant experts after several successful prompt calibrations. This will then allow the study of their common features for future automation.

4 Summary

The first automation of the Belle II detector's prompt calibration processing has been achieved on 2019 data through the use of a custom plugin to the Apache Airflow package. A globally available website for detector experts to submit CAF jobs and collect the results has been created. Integration with the existing Belle II Jira ticketing system has also been well received, allowing communication between experts without introducing a new unfamiliar technology. The main development for the future will be to include more automated validation and monitoring of calibration jobs.

References

[1] Belle II Collaboration, *Measurement of the integrated luminosity of the Phase 2 data of the Belle II experiment*, arXiv:1910.05365 [hep-ex] (2019)

[2] Tetsuo Abe *et al.*, *Belle II Technical Design Report*, arXiv:1011.0352 [physics.ins-det] (2019)

[3] Tadeas Bilka *et al.*, *Alignment and Calibration of the Belle II Detector*, EPJ Web of Conferences **214** (2019)

[4] Thomas Kuhr *et al.*, *The Belle II Core Software*, Comput.Softw.Big Sci. **3** 1 (2019)

[5] David Dossett *et al.*, *Status of the calibration and alignment framework at the Belle II experiment*, J. Phys. Conf. Ser. **898** (2017)

[6] Gianluca Cerminara and Broen van Besien, *Automated workflows for critical time-dependent calibrations at the CMS experiment*, J. Phys. Conf. Ser. **664** (2015)

[7] Maurizio Martinelli, *Novel Real-time Alignment and Calibration of the LHCb detector in Run2*, J. Phys. Conf. Ser. **898** (2017)

[8] Lynn Wood, Marko Bracko, Todd Elsethagen, Kevin Fox, Carlos Gamboa, Thomas Kuhr, Martin Ritter, *Performance of the Belle II Conditions Database*, EPJ Web of Conferences **214** (2019)

[9] Airflow, https://airflow.apache.org/

[10] Celery, https://www.celeryproject.org/

[11] SQLAlchemy, https://www.sqlalchemy.org/

[12] Jakob Blomer *et al.*, *The CernVM File System: v2.5.1* https://doi.org/10.5281/zenodo.1472994 (2018)

[13] Federico Stagni *et al.*, *DIRACGrid/DIRAC: v6r20p15*, https://doi.org/10.5281/zenodo.1451647 (2018)