Automating Calibration at the Belle II Detector

D. Dossett and M. Sevior (University of Melbourne)
david.dossett@unimelb.edu.au

Belle II aims collect a data sample 50 times larger than the previous generation of B-Factories taking advantage of the SuperKEKB design luminosity of 8×10^{35} cm^{-2} s^{-1}. As Belle II moves forwards, the new data will need to be calibrated promptly before first reconstruction can proceed. Collation of valid calibration data, submission of calibration jobs, validation, and upload to the conditions database will all require automation and monitoring. The **b2cal** package is under development to achieve this automation, using the **Apache Airflow** project as a base.

Belle II Calibration and Alignment Framework (CAF)

What it can do*	What it can't do
Configure and run multiple calibrations	Tell you which data files to use in your processes
Define dependencies between calibrations	Provide a monitoring UI
Parallelise over input files by submitting to batch system backends	Run validation jobs
Configure exactly how an algorithm runs over data using plugin classes	Extend to run other data production jobs prior to calibration
Restart from last successful point after a crash	Co-ordinate a central production environment

*For more information on the CAF, see Tadeas Bilka's presentation (#467) in online computing, Track 1

We require software to organise and monitor CAF processes. And to connect with the rest of the Belle II data production

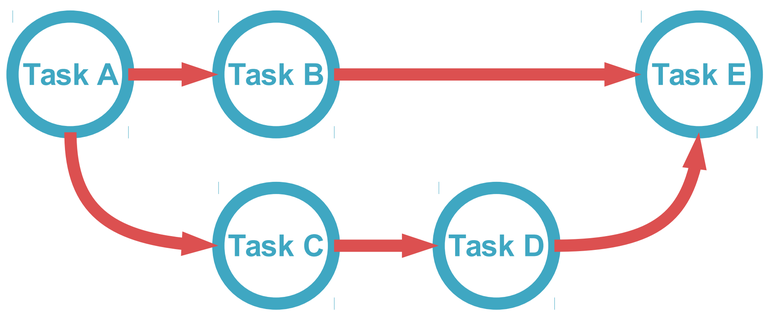
Workflow Management Systems (WMS)

- Ability to define dependencies between different tasks and scheduling them is key.
- Gaining popularity as businesses increasingly have many complex Extract, Transform, Load (**ETL**) operations to schedule e.g. search indexing/ranking, monitoring statistics creation.
- Several projects have gained prominence in HEP and commercial sector for different use cases, see below.

Some examples of projects that contain different WMS solutions. Clockwise from top: Airflow[1], Lightflow[2], Makeflow[3], DIRAC[4], and Luigi[5].

Apache Airflow

- Originally developed for Airbnb, currently moved to the Apache Software Foundation as an Apache Incubator project.
- Python scripting to define Directed Acyclic Graphs (**DAGs**) as collections of tasks (**Operators**) with dependencies between them.

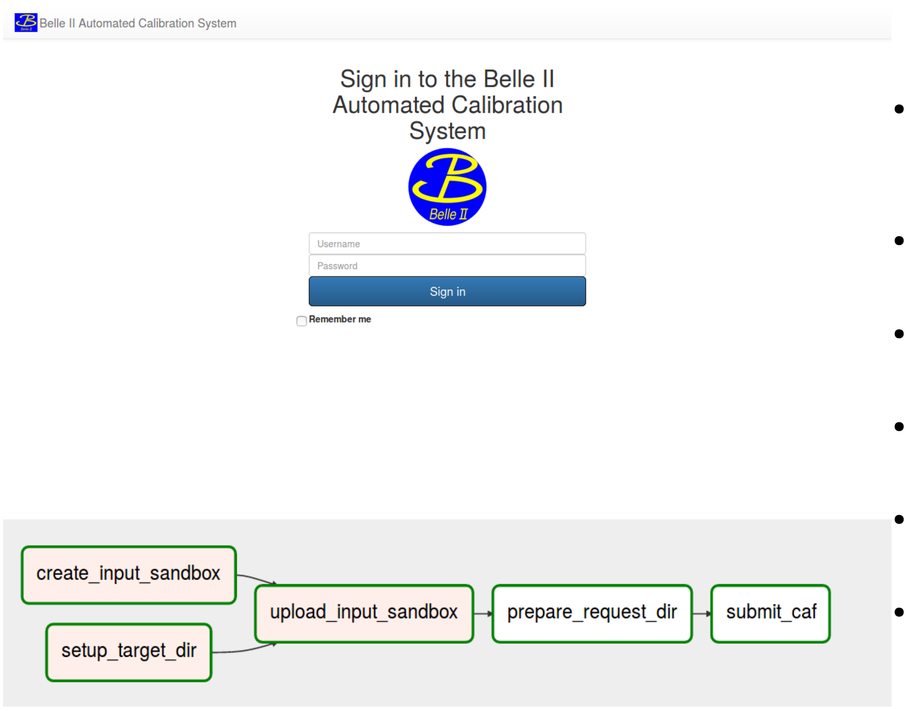


Simple diagram of a DAG representation of tasks. Vertices are tasks and edges are the dependencies between them.

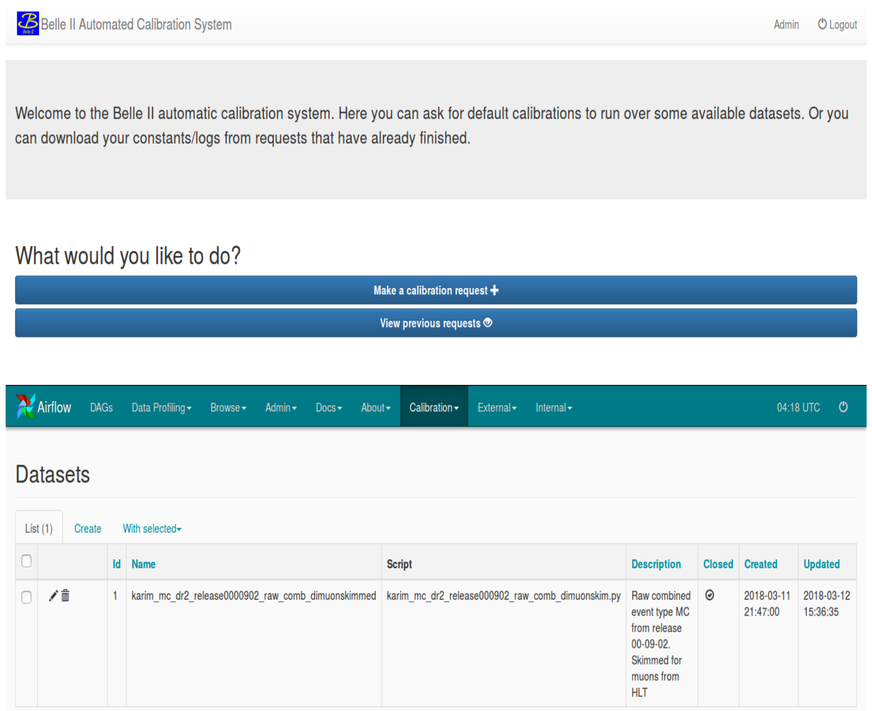
- DAGs can be given a regular schedule, triggered manually, or even trigger each other.
- Includes **Flask** based web monitoring using the **flask-admin** package.
- Uses **Jinja2** templating and **SQLAlchemy** to write tasks that render templated strings e.g. in database query strings or bash scripts.

Operator	Sensor	Hook
Defines the task in a DAG	Operator that periodically executes a query	Defines the interface to some external system
Different operator types e.g. Python, Bash, SQL	Won't complete until conditions are met	Retrieves authentication stored in Airflow DB

The b2cal Package



- Uses Airflow's plugin features to add Operators, Sensors, and Hooks for connecting to KEKCC and the Belle II DIRAC servers.
- Forked Airflow to alter the main Flask app and add role based access.
- Developing a new Flask blueprint to create a user interface for data managers.
- SQLAlchemy models for Datasets, LFNs, Calibration Requests, etc.
- Developing the various DAGs to run CAF processes.
- Easy to install with pip and get the webserver running locally in minutes!



1. https://airflow.apache.org/ 2. https://australiansynchrotron.github.io/lightflow/ 3. http://ccl.cse.nd.edu/software/makeflow/ 4. http://diracgrid.org/ 5. https://github.com/spotify/luigi