Status of the Calibration and Alignment Framework at the Belle II Experiment

David Dossett
University of Melbourne

Belle II

CHEP 2016
Belle II Experiment

• SuperKEKB $e^+e^-$ collider at KEK Tsukuba, Japan

• Will provide 40x the luminosity of KEKB

• Running mainly at upsilon 4S resonance to produce $\sim10^{10}$ $B\bar{B}$ pairs/year

• Start of full detector collisions planned for 2018
Calibration at Belle II

- What we need to do (the basics):
  - Support experts with tools to create algorithms and database payloads
  - Create an automated system to produce/upload/monitor constants for prompt reconstruction
C++ Framework

- **C++ classes built upon the Belle II analysis software framework (basf2)**

- **Collector Module:**
  - Produces ntuples/histograms/etc from Belle II event data
  - Data separated into runs
  - Used as input to algorithm step

- **Algorithm Class:**
  - Reads collected data for a range of Intervals of Validity (IoV) defined by user
  - Produces constants and saves to local database payloads
  - Can request iteration of collector step
C++ Framework

• Algorithm developers only need to create two classes:

Collector Module: Developer overrides prepare() and collect() functions

Algorithm Class: Developer overrides calibrate() function
Example: VXD Alignment Effect on Physics

- Sub 5µm and 0.06mrad systematics already achieved
- No significant effect on measurements sensitive to alignment

Ideal Geometry

10µm misalignment 0.2mrad twist
30µm misalignment 0.6 mrad twist
Python Framework

- Created a user friendly interface that developers can call from their normal steering files.

- **Some Features:**
  - Setup as many calibrations (collector + algorithms) as you want, with input data files
  - Define dependencies of calibrations on one another → pass constants to each subsequent step
  - Can submit collector step to local multiprocessing or batch systems to parallelise
  - If an algorithm doesn't have enough data in a run it will merge with the next and re-run.
  - Automatic iteration if requested + passing constants to new iteration

- **Should allow the future automation framework and developers to run nearly the same code**
Example code 1

• Simplest use case: One calibration, run locally

```python
# import statements
...

alg = TestAlgorithm()

cal = Calibration(name = "Test1",
    collector = "collector_module_name",
    algorithms = [alg],
    input_files = ["/path/to/test1.root",...])

fw = CAF()

fw.add_calibration(cal)

fw.run()
```
Example code 2

- Multiple calibrations, dependencies and PBS backend

```python
...  
cal_b.depends_on(cal_a)  # cal_a MUST complete before cal_b
fw = CAF()
fw.add_calibration(cal_a)
fw.add_calibration(cal_b)
fw.backend = PBS()  # Configurable if needed
fw.run()
```
simple python steering file

```python
#!/usr/bin/env python3
fw = CAF()
...
```

- Finite State Machines (FSMs) enforce safe movement to new states
- Threads can check state of other FSMs

- Calibration A Runner Thread
  - Calibration A FSM

- Calibration B Runner Thread
  - Calibration B FSM

- Calibration C Runner Thread
  - Calibration C FSM

Submission of collector or algorithm jobs on input data

Backend Wrapper e.g. Local subprocess, PBS, LSF, ...
FSM Framework

- Moved to an explicit FSM framework to reduce code complexity in flow logic
- Nicely defines transitions between states
- Adding new states/transitions is very easy
- Potential to allow 'checkpoints' so that a calibration can be restarted from an intermediate step

Easy to visualise logic

![FSM Diagram]

- Running algorithms
- Algorithms completed
- Completed
- Finish
- Iterate
- Init
- Running collector
- Collect
- Fail
- Complete
- Collector completed
- Collector failed
- Submit collector
- Algorithms failed
- Fail
- Complete
Outlook

• C++ and Python frameworks are quite mature and easy for developers to use, even with large quantities of data or interdependencies.

• Some algorithms already in the framework:
  - Millepede for alignment of VXD
  - PXD Cluster Shape
  - EKLM Time Calibration

• Supporting detector experts while they develop new algorithms

• Automating/monitoring constants production will now be the focus