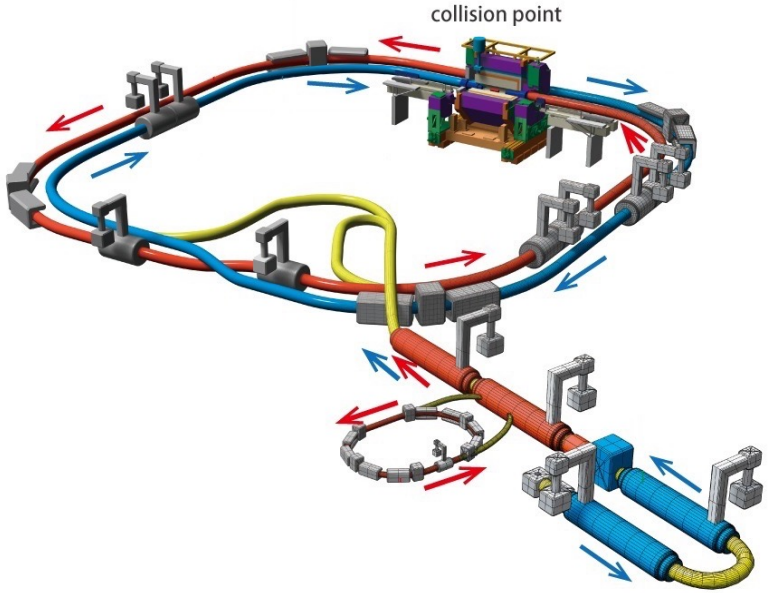
Python in the Belle II experiment

Yo Sato (KEK, IPNS)

Contents

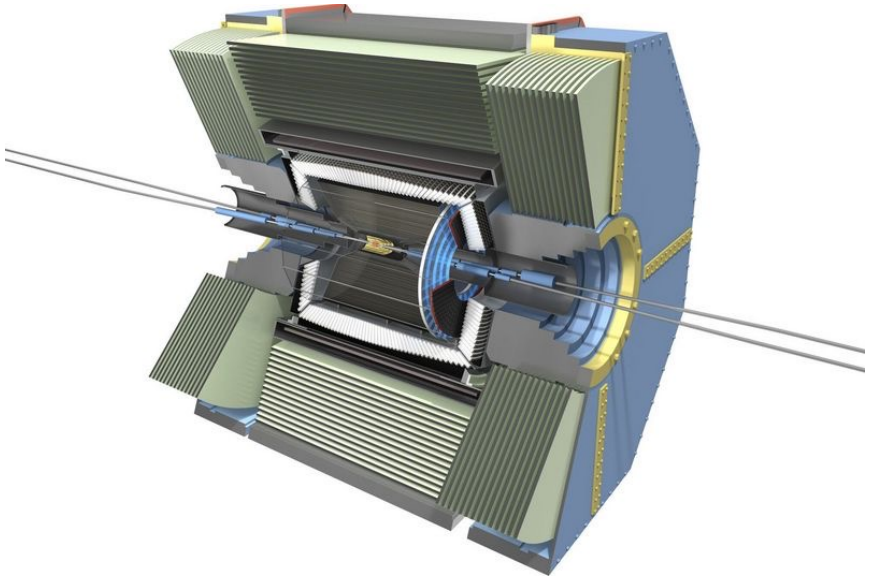
- Introduction of the Belle II experiment
- Python in the Belle II Analysis Software Framework
- Python for workflow management at Belle II

SuperKEKB / Belle II



SuperKEKB

- Asymmetric e^- (7 GeV) e^+ (4 GeV) collider.
- CM-energy is at $\Upsilon(4S)$ resonance, 10.58 GeV.
  ⇒ Produce $B\bar{B}$-pair efficiently.
- Goals: $\sim 6\times10^{35}\ \mathrm{cm^{-2}s^{-1}}$, $50\ \mathrm{ab^{-1}}$

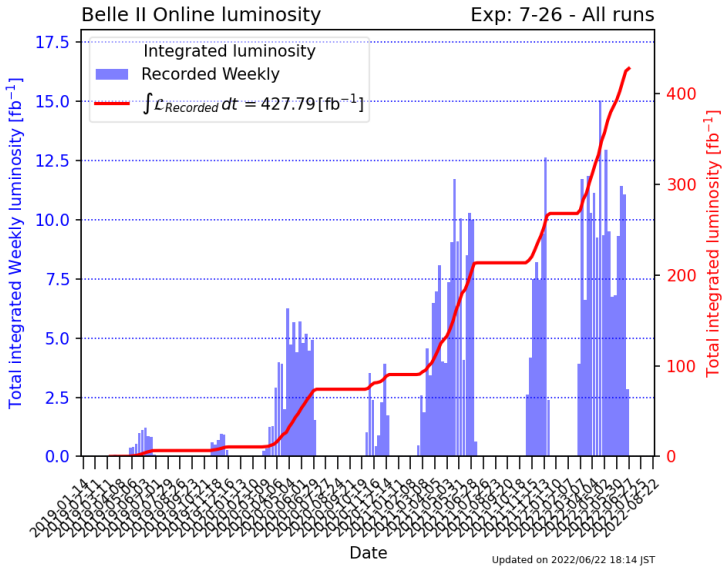


Belle II detector

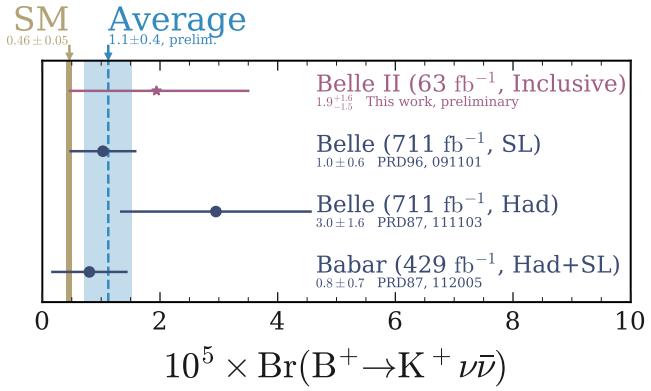
- Upgraded from the Belle detector;
  - Improved vertex resolution
  - Enhanced K/π separation
  - New trigger lines
  - ... and more

Status of the Belle II experiment

- Total integrated luminosity: $\sim 430\ \mathrm{fb}^{-1}$
- World record of luminosity: $4.71\times10^{34}\ \mathrm{cm}^{-2}\mathrm{s}^{-1}$
- Belle II started to produce high-quality results with the upgraded detector and analysis techniques.

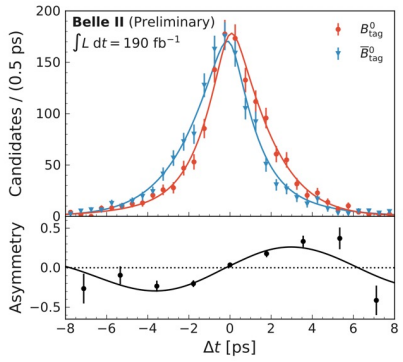


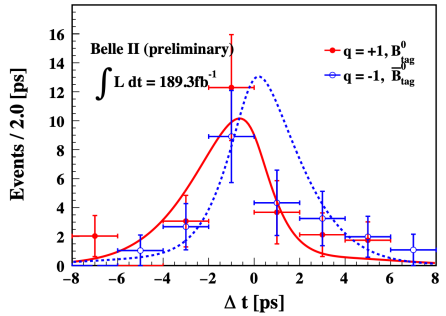
$B^+ \to K^+\nu\bar{\nu}$ search



Phys.Rev.Lett. 127 (2021) 18, 181802

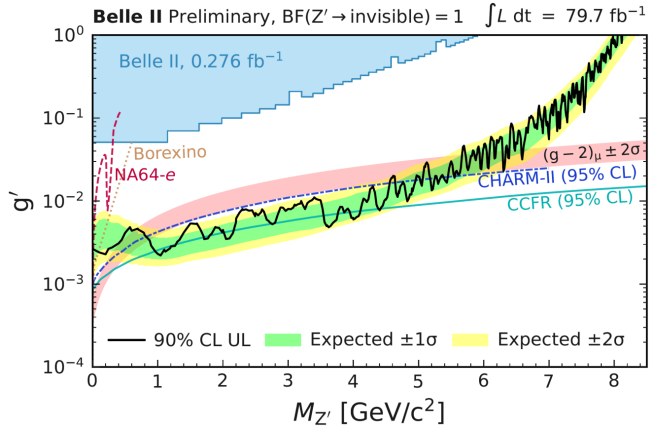
B-meson time-depended CP-violation





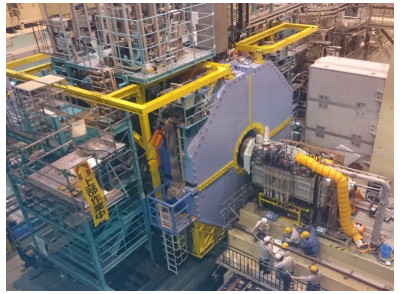
C. La Licata @ ICHEP 2022

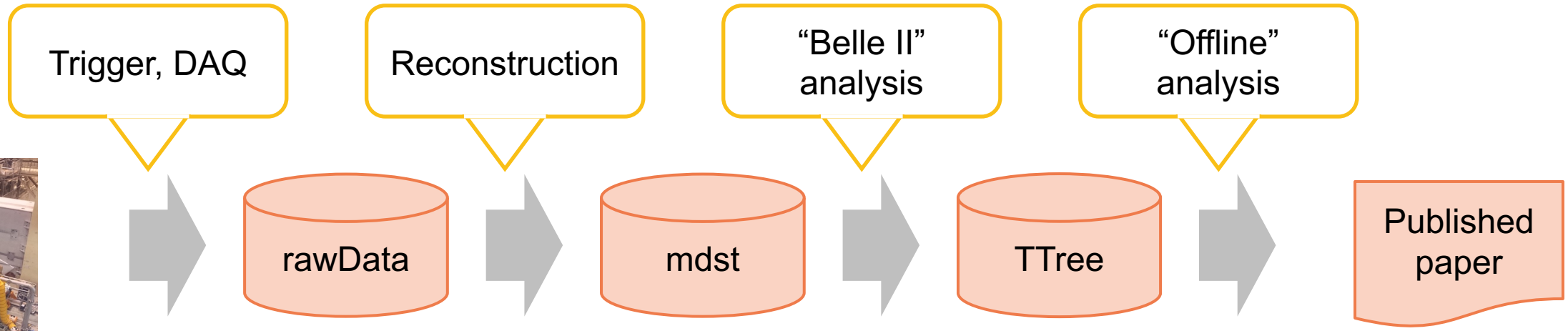
$Z' \to$ invisible search



E. Graziani @ ICHEP 2022

Belle II Analysis Software Framework





e.g.
Detector-readout

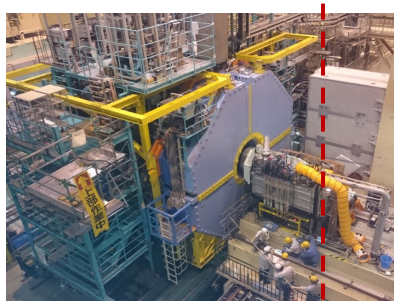
e.g.
Track,
Calorimeter-cluster

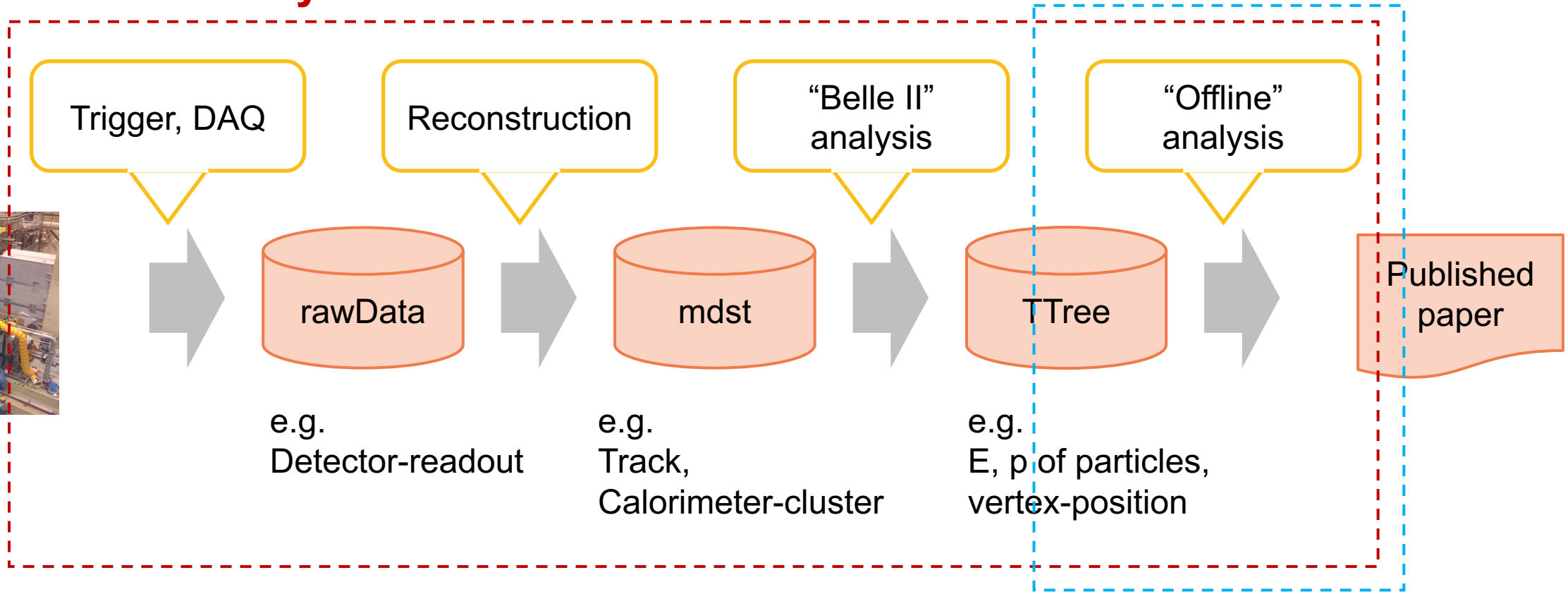
e.g.
E, p of particles,
vertex-position

mdst
= mini-data-summary-table

Belle II Analysis Software Framework

Belle II Analysis Software Framework: `basf2`

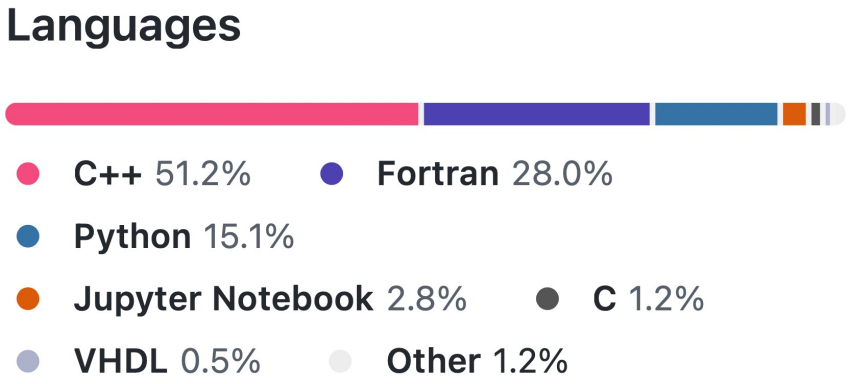




- The Belle II software, basf2, is completely new and is used for everything from
  - triggering data,
  - generation of Monte Carlo events,
  - tracking, clustering,
  - high-level analysis.

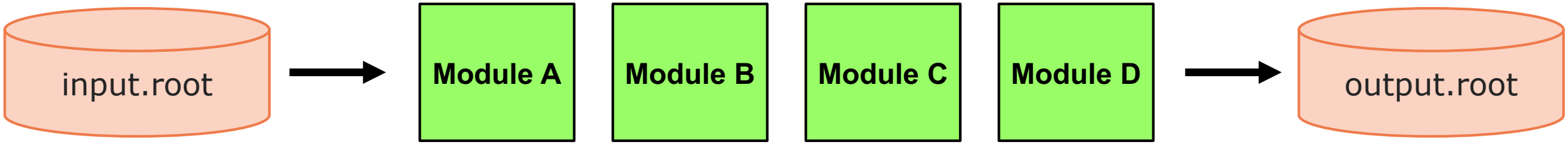
Comput.Softw.Big Sci. 3 (2019) 1, 1

- Open-source software since 2021 (https://github.com/belle2/basf2)
- ROOT for I/O, C++ for heavy lifting, Python for steering script.



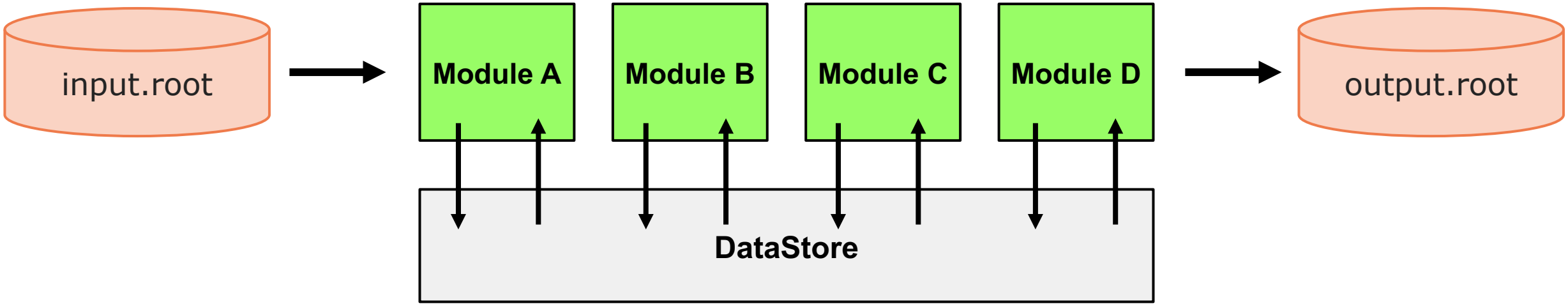
Design of basf2 analysis

- A set of classes that process the data
  ⇒ **basf2 Module**



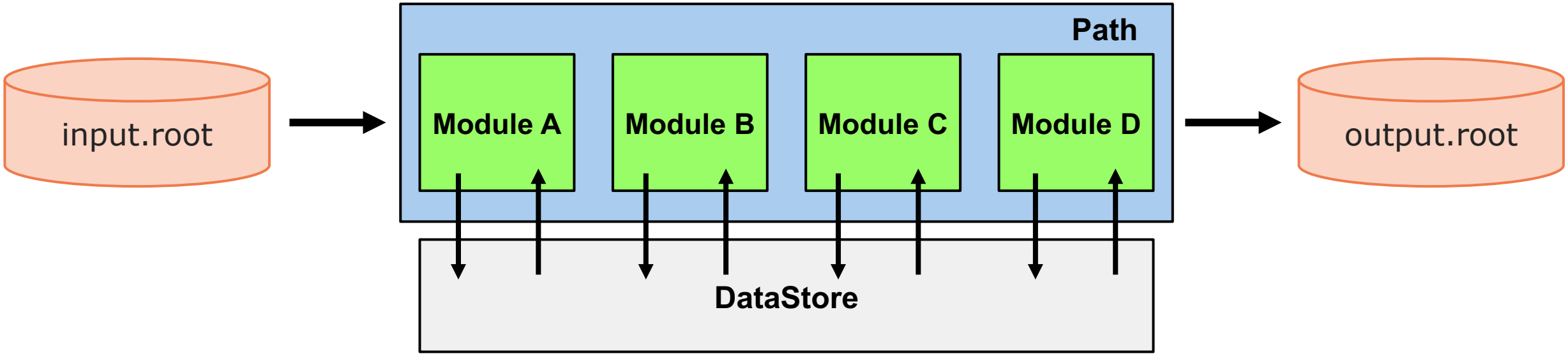
Design of basf2 analysis

- A set of classes that process the data
  ⇒ **basf2 Module**
- A set of classes that hold the data and allow modules to pass thing one to the other
  ⇒ **basf2 DataStore**

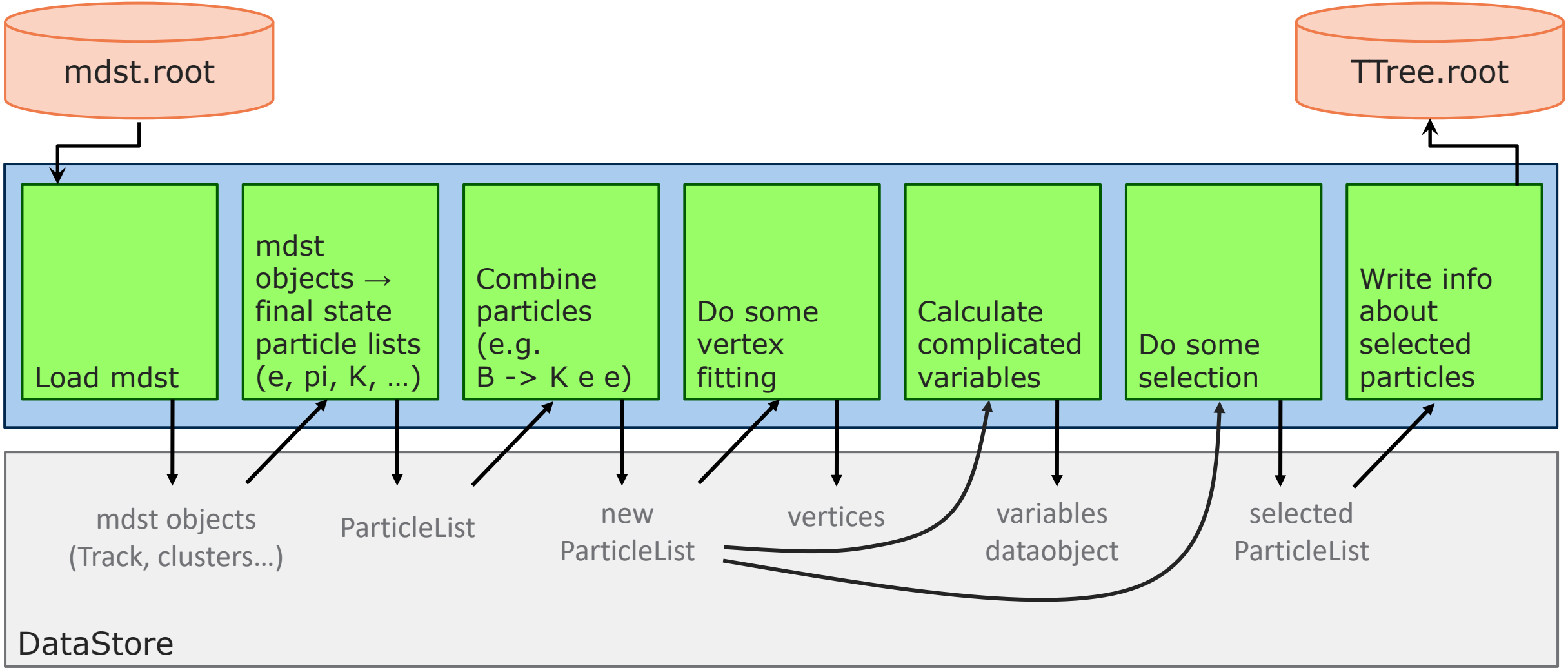


Design of basf2 analysis

- A set of classes that process the data
  ⇒ **basf2 Module**
- A set of classes that hold the data and allow modules to pass thing one to the other
  ⇒ **basf2 DataStore**
- An order in which the modules must be executed
  ⇒ **basf2 Path**



Typical path for an analysis job



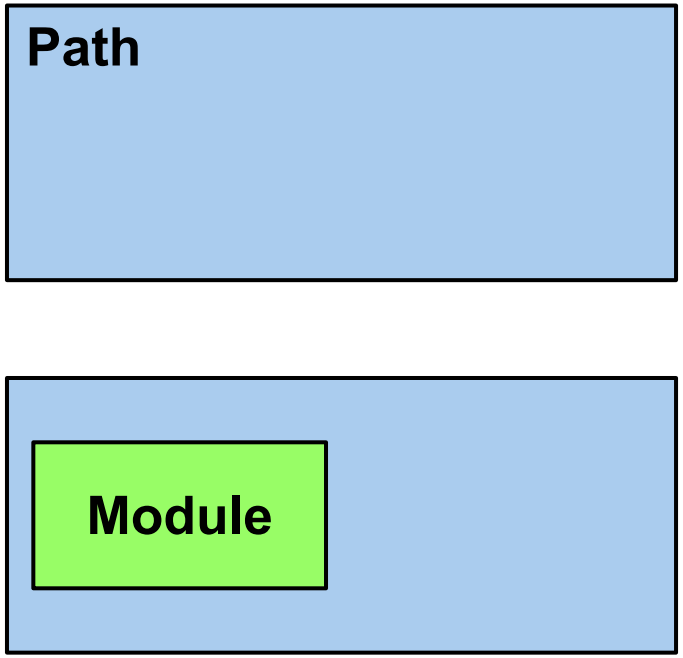
Python usage in the basf2

- Steering file to arrange appropriate *modules* into a *path*, configure *modules'* options, and start data processing.
- Easy-understanding syntax for new users.

- basf2 *modules* can be written in C++ and Python.
  - Framework modules are written in compiled C++ codes.
- Functionality is exported via `boost::python`.
- Access to the *DataStore* is provided with PyROOT.
- Documentation web page is generated with Sphinx.

Python steering file


```
import basf2

# create a path
mypath = basf2.Path()

# Add modules in the path
# input mdst file
mypath.add_module('RootInput',
                  inputFileName='mdst.root')

# Put your modules

# process the events
basf2.process(mypath)
```



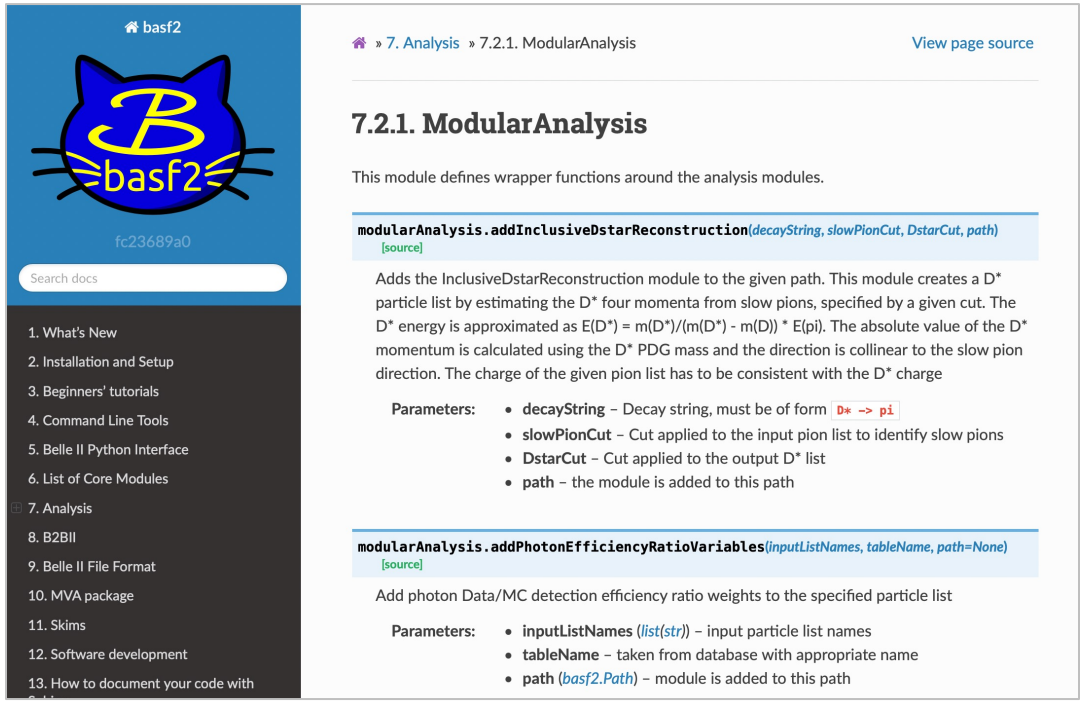
Modules are executed for each event with the given order of *path*.

Python steering file

- Wrapper functions for modules configuration are provided with well-documented Sphinx web page.

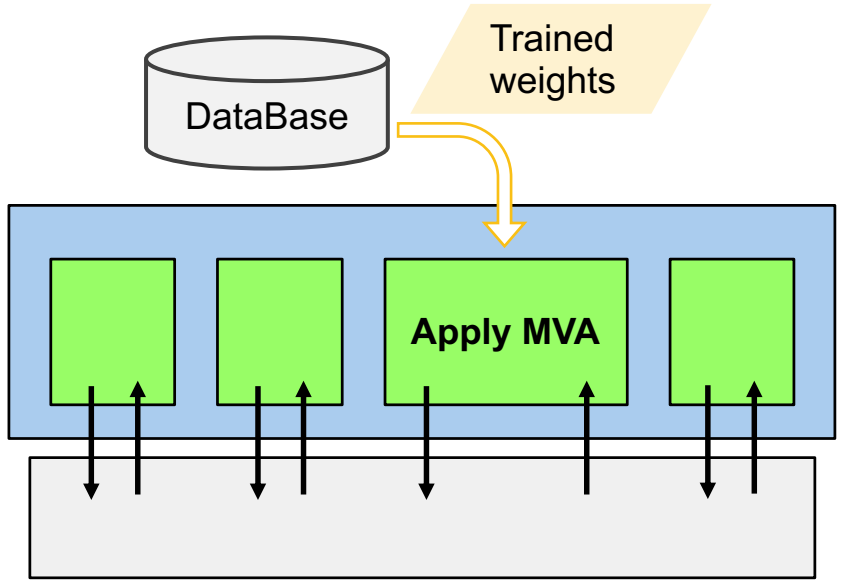
https://github.com/belle2/basf2/blob/main/analysis/scripts/modularAnalysis.py

```
def matchMCTruth(list_name, path):
    """
    Performs MC matching (sets relation Particle->MCParticle) for
    all particles (and its (grand)^N-daughter particles) in the specified
    ParticleList.

    @param list_name name of the input ParticleList
    @param path      modules are added to this path
    """

    mcMatch = register_module('MCMatcherParticles')
    mcMatch.set_name('MCMatch_' + list_name)
    mcMatch.param('listName', list_name)
    path.add_module(mcMatch)


def reconstructDecay(decayString,
                     cut,
                     dmID=0,
                     writeOut=False,
                     path=None,
                     candidate_limit=None,
                     ignoreIfTooManyCandidates=True,
                     chargeConjugation=True,
                     allowChargeViolation=False):
    r"""
    Creates new Particles by making combinations of existing Particles - it reconstructs unstable particles via their specified
    decay mode, e.g. in form of a :ref:`DecayString`: :code:`D0 -> K- pi+` or :code:`B+ -> anti-D0 pi+`, ... All possible
    combinations are created (particles are used only once per candidate) and combinations that pass the specified selection
    criteria are saved to a newly created (mother) ParticleList. By default the charge conjugated decay is reconstructed as well
    (meaning that the charge conjugated mother list is created as well) but this can be deactivated.
```



MVA/ML in basf2

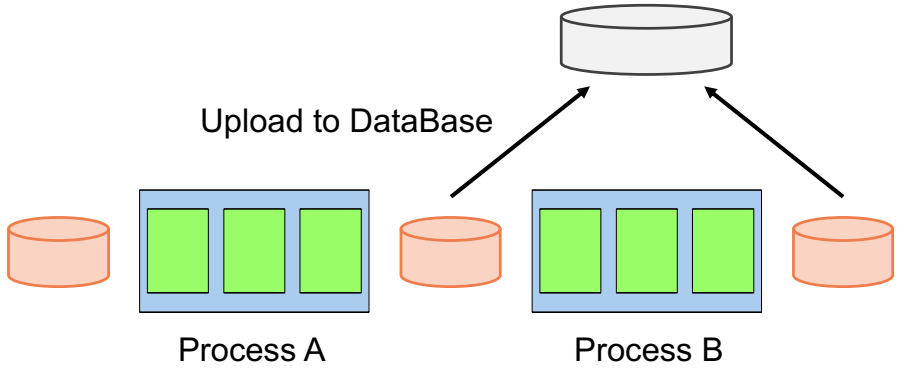
- The mva package of basf2 provides tools to integrate mva/ml method.
  - Full Event Interpretation (Comput.Softw.Big Sci. 3 (2019) 1, 6) and Flavor Tagger (Eur.Phys.J.C 82 (2022) 4, 283) are deployed with the mva package.
  - Default method is the FastBDT (Comput.Softw.Big Sci. 1 (2017) 1, 2)
  - Handy tools for training and validation are provided
- Frameworks that have a Python interface are supported, e.g. TensorFlow, Keras, XGBoost.



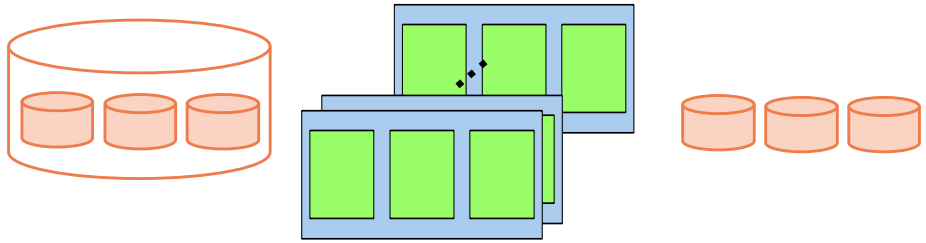
Python for workflow management at Belle II

Complex workflow has to be managed time-effectively and with good reproducibility.

- Data calibration
  - Interdependent steps to deal with the detector conditions that can vary over time.
  - To provide results in a timely manner, the prompt calibration of data is necessary



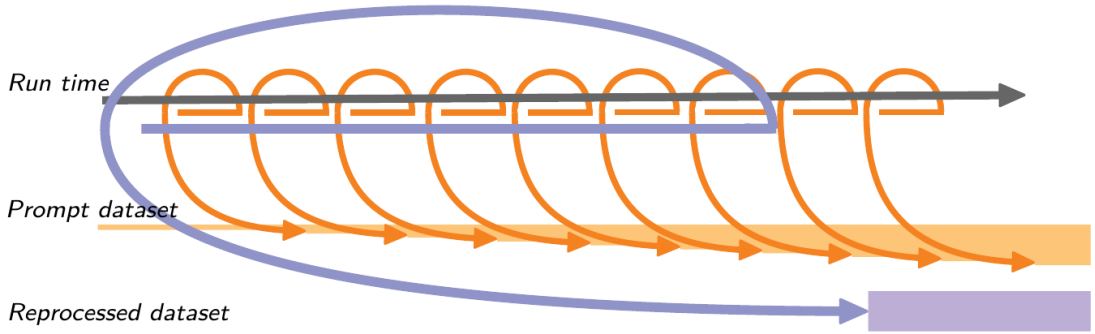
- Analysis with new data and new software version
  - Reproduce the analysis procedures for new data
  - Thousands of batch jobs have to be managed



Python plugin packages are developed and used for the workflow management.

Calibration strategy at Belle II data

- Reprocessing
  - Make a full recalibration. Every 6-12 months
- Prompt calibration
  - Initial processing of recently collected data
  - Every a couple of weeks

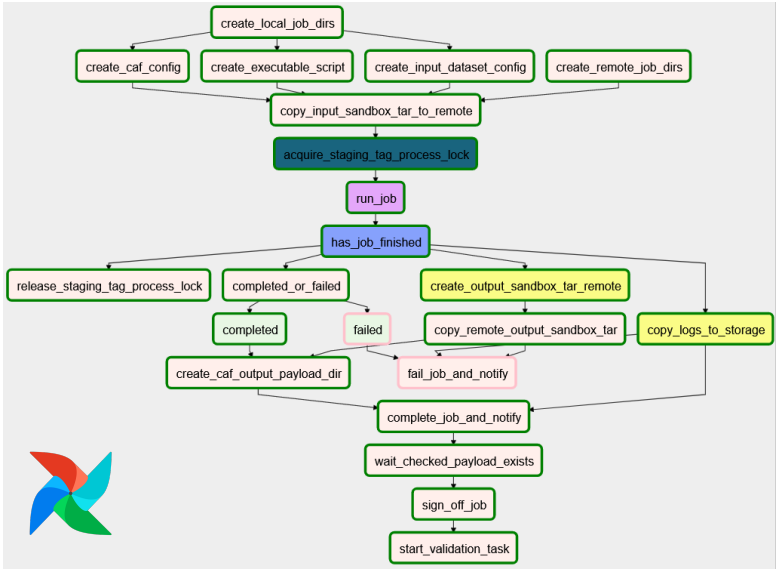


b2cal (*EPJ Web Conf.* 245 (2020) 02016)

- Based on the Apache Airflow.
- Developed to automate the prompt calibration.

Airflow based workflow management : b2cal

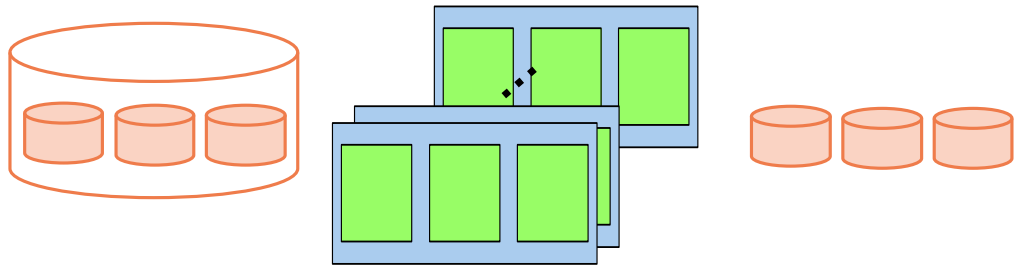
- Calibration has a very complex workflow composed of interdependent tasks
  - submit jobs, open/close Jira tickets, data registration to the DataBase, …
- Apache Airflow
  - Directed Acyclic Graph (DAG) workflow manager
  - Good web monitoring of tasks and submitting jobs
- Automation reduced the possibility of human errors.
- Time taken to perform the prompt calibration is getting decreased as more experience is gained.



EPJ Web Conf. 251 (2021) 03019

Analysis reproducibility

- Reproducibility of analysis for new dataset and new basf2 version.
- Analysis has huge number of batch jobs for segmentalized input data.



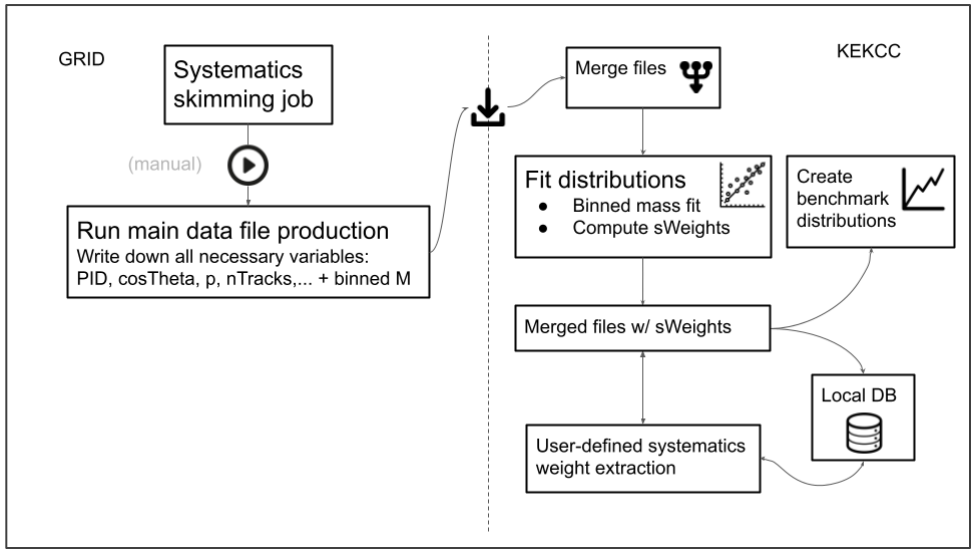
b2luigi (https://github.com/nils-braun/b2luigi)

- Based on luigi
- Helper package for luigi with thousands of batch jobs = Batch 2 luigi
- Originally developed for Belle II
  - basf2 helper functions, Submission of GRID jobs

Systematic Corrections Framework at Belle II

- Performance study of the particle identification is managed by the Systematic Corrections Framework. (https://syscorrfw.readthedocs.io)
  - Difference between data and MC is a main source of systematic uncertainty for many physics analyses.
  - Reproduce the performance study for new data and MC.
  - Largely inspired by LHCb PIDCalib framework

- The developed library is based on b2luigi.
  - basf2 analysis jobs for control samples
  - Perform a fit on distributions
  - Produce weights of ratio between data/MC



Summary

- Python is a key language in the Belle II experiment thanks to the user-friendly interface and extensibility to many libraries.

- Python is a first-class citizen in the Belle II Analysis Software Framework, basf2.
  - Steering script to arrange modules into a path with easy-understanding syntax
  - MVA/ML methods with a Python interface are supported for basf2 analysis

- Complex workflows of Belle II software are managed with Python packages.
  - b2cal: Prompt calibration tools based on the Apache Airflow
  - b2luigi: Analysis reproducibility with thousands of batch jobs based on luigi