

# The Belle II Analysis Software Framework

Frank Meier

software coordinator of the Belle II collaboration

The 2024 International Workshop on Future Tau Charm Facilities  
University of Science and Technology of China  
14-18 January 2024



Research supported by



U.S. DEPARTMENT OF  
**ENERGY**

Office of  
Science

# Introduction

- ▶ Belle II Analysis Software Framework (basf2)
  - ▶ source code publicly available at <https://github.com/belle2/basf2>
  - ▶ documentation publicly available at <https://software.belle2.org>
- ▶ basf2 links against defined set of third-party libraries that we call **externals**
  - ▶ publicly available at <https://github.com/belle2/externals>
- ▶ repository with scripts to install and set up basf2 called **tools**
  - ▶ publicly available at <https://github.com/belle2/tools>
- ▶ repository with script for version managing (recommended releases and global tags)
  - ▶ publicly available at <https://github.com/belle2/versioning>
- ▶ LGPL (GNU Lesser General Public License) version 3 or later
  - ▶ header in each file:

basf2 (Belle II Analysis Software Framework)  
Author: The Belle II Collaboration

See git log for contributors and copyright holders.  
This file is licensed under LGPL-3.0, see LICENSE.md.

# Externals

- ▶ versioned set of external software packages used and linked against the Belle II software
- ▶ dependency between packages considered and compatibility guaranteed
- ▶ C++ packages like
  - ▶ ROOT, XRootD
  - ▶ gcc, clang, gdb, cmake, Python
  - ▶ boost, Eigen, gsl
  - ▶ EvtGen, Geant4, clhep, PYTHIA
  - ▶ git, cppcheck, doxygen
- ▶ includes patches
- ▶ python packages like pandas, matplotlib, torch, tensorflow, jupyter, ...
- ▶ source files uploaded to web server to never lose availability

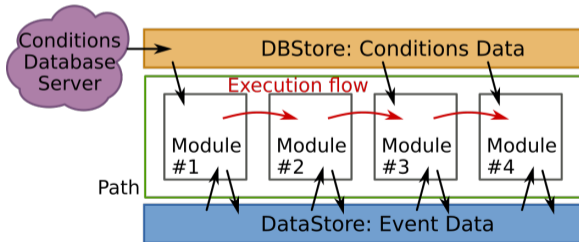
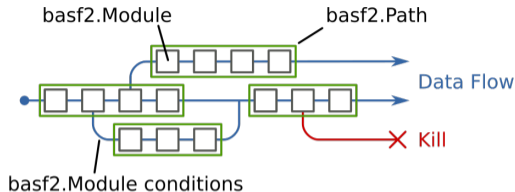
# Tools

- ▶ collection of scripts to prepare environment for execution of Belle II software
- ▶ b2setup
  - ▶ setting environment variables
- ▶ b2code-create, b2code-style-check, b2code-style-fix, b2code-clean
  - ▶ creating local directory for core software development and fixing style issues
- ▶ b2install-prepare, b2install-release, b2install-externals, b2install-data
  - ▶ installing pre-compiled software versions or example data on local machine
- ▶ b2analysis-create, b2analysis-get, b2analysis-update
  - ▶ creating local directory for development of analysis code including preparation of build system and addition of repository to git



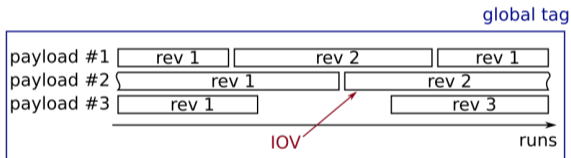
# Modular structure

- ▶ linear arrangement of C++ modules in a path
- ▶ core functions of modules
  - ▶ initialize
  - ▶ beginRun
  - ▶ event
  - ▶ endRun
  - ▶ terminate
- ▶ python steering script to set up path



# Conditions Database

- ▶ storage place of additional data needed to interpret and analyze the data that can change over time, e.g., detector configuration or calibration constants
- ▶ payloads: binary objects (usually ROOT files) identified by name and revision number
- ▶ each payload has defined intervals of validity (iovs), i.e., experiment and run range
- ▶ globaltag: collection of payloads and iovs for a certain dataset, identified by unique name



- ▶ once prepared globaltag is immutable and cannot be modified any further to ensure reproducibility of analyses
- ▶ different processing iterations use different globaltags
- ▶ globaltag of reconstruction stored in metadata and automatically applied
- ▶ chain of globaltags possible

# Data-taking

- ▶ basf2 runs on 12 high-level trigger nodes
- ▶ ZeroMQ
  - ▶ acts like concurrency framework while looking like an embeddable networking library
  - ▶ sockets that carry atomic messages across various transports like in-process, inter-process, TCP, and multicast
  - ▶ fast
  - ▶ asynchronous I/O model → scalable to multi-core operation



# Reconstruction chain

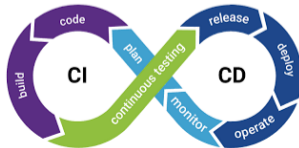
- ▶ RootInputModule
- ▶ geometry
- ▶ clustering of calorimeter
- ▶ clustering of pixel and silicon vertex detectors
- ▶ track finding
- ▶ track fitting
- ▶ track extrapolation
- ▶ track-cluster matching
- ▶ software trigger
- ▶ post-filter tracking
- ▶ PID
- ▶ RootOutputModule

# Tracking

- ▶ pattern recognition / track finding
  - ▶ finding hits belonging to the same charged particle
  - ▶ SectorMaps → Segment Network → Cellular Automaton to find longest paths
  - ▶ inter-detector track finding via Combinatorial Kalman Filter
- ▶ track fit
  - ▶ extracting track parameters from fit to collection of hits
  - ▶ Deterministic Annealing Filter (DAF)
  - ▶ in Belle II currently use GenFit package ([DOI:10.5281/zenodo.10301439](https://doi.org/10.5281/zenodo.10301439))

# Unit-tests

- ▶ first layer of software validation
- ▶ run full test suite for each commit of open merge requests and each merge into main or release branch
- ▶ currently about 1000 unit-tests of C++ code using GoogleTest
  - ▶ check basic functionality of modules, return values of functions and variables
- ▶ about 300 additional python tests
  - ▶ make sure that standard scripts do not crash
  - ▶ compare output of certain scripts with reference expectation, e.g. for mdst backward compatibility
- ▶ unit-tests intended to catch non-trivial dependencies and implications of code changes
- ▶ running all tests (in 16 parallel processes) takes 15-20 min



## Nightly validation

- ▶ run once per day (night)
- ▶ workflow of nightly validation
  1. generate smallish samples
  2. run validation scripts
  3. create output histograms
  4. comparison with reference
    - ▶ calculate p value of histogram compatibility
    - ▶ calculate performance numbers, *e.g.*, width of distribution
- ▶ plots of various software releases uploaded to web server
- ▶ email notifications sent out to assigned contacts

# Nightly validation

 Compact View  Expert Content

**Belle II**  
Validation

Revisions

Popular/prebuilt/default:

**reference**

- upgrade-2023-08-15
- 2023-11-29 00:58 JST | [ec00bd1bc](#)
- nightly-2024-01-12**
- 2024-01-12 13:02 JST | [03a05fd8](#)
- nightly-2024-01-11
- 2024-01-11 14:37 JST | [469715224](#)
- nightly-2024-01-10
- 2024-01-10 13:17 JST | [9150ba89e](#)
- prerelease-08-00-00b
- 2023-07-18 00:40 JST | [3cb0bebeb](#)
- prerelease-08-00-00a
- 2023-07-04 07:17 JST | [1149e4fe1](#)
- prerelease-07-00-00d
- 2022-12-04 20:18 JST | [a21fe6464](#)
- release-08-00-04**
- 2024-01-11 21:41 JST | [9eb4a8b4d](#)
- release-08-00-03
- 2023-12-16 14:46 JST | [b3593e1ef](#)
- release-08-00-02
- 2023-12-09 22:07 JST | [698a3f10f](#)
- release-08-00-01
- 2023-10-31 20:14 JST | [e006da723](#)
- release-08-00-00
- 2023-10-22 13:54 JST | [041214285](#)
- release-08-00
- 2023-12-16 07:43 JST | [b3593e1ef](#)
- release-07-00
- 2023-06-23 14:40 JST | [df1f0385a](#)
- release-06-02-00
- 2023-12-28 06:27 JST | [e2aa77080](#)
- release-06-01-15
- 2023-12-09 05:10 JST | [16525535](#)

## Package: svd

### Script Files

[nightly-2024-01-12](#) [release-08-00-04](#)

Failed Scripts

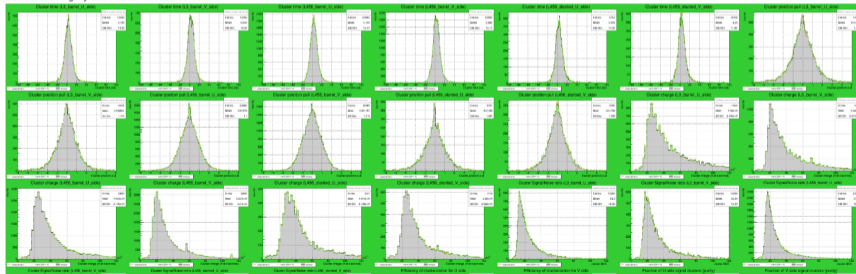
No failed scripts

Finished Scripts

Skipped Scripts

### Result File: SVDClusterPerformance

Downloads: [nightly-2024-01-12](#) [release-08-00-04](#)



# Monitoring

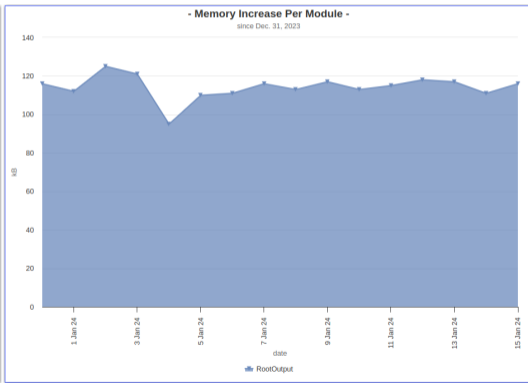
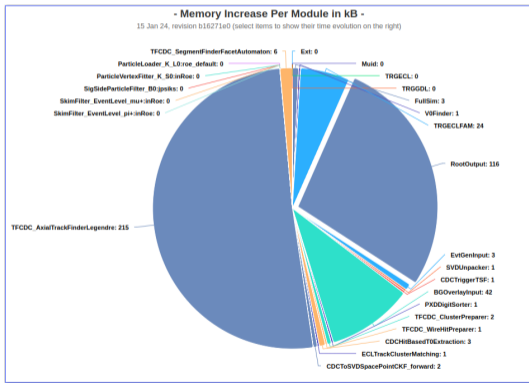
- ▶ nightly build run with different configurations (debug, intel, clang)
- ▶ many resource checks (memory consumption, execution time, output file size)
  - ▶ summarize build warnings, cppcheck, doxygen check, dependency check, geometry overlap check

The screenshot shows a table titled "Results of development build" with the following columns: Package details, Version, and a series of columns representing different build configurations (e.g., Debug, Intel, Clang) and various checks (e.g., warnings, errors, memory). Each cell in the table contains a status icon (green checkmark, red warning triangle, or red error icon) and a numerical value.

- ▶ history plots of warning and error counters as well as resource usage

# Monitoring

- ▶ nightly build run with different configurations (debug, intel, clang)
- ▶ many resource checks (memory consumption, execution time, output file size)
  - ▶ summarize build warnings, cppcheck, doxygen check, dependency check, geometry overlap check



- ▶ history plots of warning and error counters as well as resource usage

# Documentation

- ▶ good documentation crucial to (recruiting) process of software development and maintenance
- ▶ sphinx
  - ▶ use reStructuredText
  - ▶ use sphinx's autodoc feature to conveniently create documentation based on python's docstrings
- ▶ doxygen for C++ documentation
- ▶ tests for (almost) all packages to ensure that everything is documented



basf2 development documentation

- 1. What's New
- 2. Installation and Setup
- 3. **Beginners' tutorials**
- 3.1. Welcome!
- 3.2. Fundamentals
- 3.3. Software Prerequisites
- 3.4. Working with Belle II software
- 3.5. Offline analysis
- 3.6. Data model and computing
- 3.7. Workflow Management
- 3.8. Join us
- 4. Command Line Tools
- 5. Belle II Python Interface
- 6. List of Core Modules
- 7. Analysis
- 8. ROOT
- 9. Background module
- 10. Calibration
- 11. Decay Files
- 12. The Belle II Event Display
- 13. Event Generators
- 14. Tools for validation of the Software Trigger
- 15. KLH (K<sub>L</sub><sup>0</sup> → J/ψ) and Muon Generator
- 16. Belle II File Format
- 17. MVA package
- 18. PID
- 19. Reconstruction
- 20. Simulation
- 21. Skims
- 22. SVD
- 23. Tracking
- 24. TRG
- 25. Tools for Physics Validation of the Software
- 26. Fitting training

## 3. Beginners' tutorials

This online textbook aims to help new Belle II members to get started with the software by following through a series of hands-on lessons.

📌 We want YOU to contribute to this book! ▼

### Tip

Just as there are many versions of the Belle II software, there are many versions of this documentation to match it. After all, if a new feature is added in our software, we also want to have the documentation for it.

The current version of this documentation is shown on the top left of this page, just below the logo. You can also change your version by clicking on [other versions](#).

If you are a new to all of this, we recommend you to select the recommended release version (`release-xx-xx-xx` (*recommended*)) in the above list.

You can also take a sneak peek at the most recent version of the documentation by selecting the [development version](#). However not all of the code examples might work for you yet.

The earliest release version which contains this online book is `release-00-00-12`.

### Warning

If you change the version to an earlier version than the current one, some pages (also this page!) might not exist.

### Warning

If you read the documentation of a newer version than the software that you use (in particular the development version), you might not yet be able to use some of the features shown.

### Tip

If you get stuck or have any questions to the online book material, the [#startek-workshop channel](#) in our chat is full of nice people who will provide fast help. However this is not the place for specific or very detailed questions about your own analysis.

The first lesson of this book ([Collaborative Tools](#)) will show several other places where you can get help later on. It also includes a quick tips section about the chat: [Asking a question in #startek-workshop](#).

- 3.1. Welcome!
  - 3.1.1. Collaborative Tools.
- 3.2. Fundamentals
  - 3.2.1. Introduction
  - 3.2.2. Data Taking



## Supported environments

- ▶ basf2 meant to work on any recent 64bit Linux system but only tested and binaries provided for
  - ▶ Enterprise Linux 7 or CentOS 7
  - ▶ Enterprise Linux 8 or CentOS 8
  - ▶ Enterprise Linux 9 or AlmaLinux 9
  - ▶ Ubuntu 20.04
  - ▶ Ubuntu 22.04
- ▶ basf2 distributed on cvmfs
- ▶ ARM version under development
- ▶ central Buildbot instance connected via gitlab webhooks to code changes → triggers builds on various workers

# Release policy

- ▶ major releases
  - ▶ once a year
  - ▶ very thorough validation
  - ▶ contains all software changes that are merged to the main branch
- ▶ minor releases
  - ▶ frequency: one to two per major release
  - ▶ limited amount of new features, usually for specific purpose
- ▶ patch releases
  - ▶ mostly for bug fixes, especially for data-taking and calibration
  - ▶ during data-taking synchronized with maintenance days
- ▶ light releases
  - ▶ every two months
  - ▶ for introduction of new offline data analysis features
  - ▶ contain only framework, mdst, mva, analysis, skim, geometry, online\_book, and b2bii packages
  - ▶ no unpacking or digitization  $\Rightarrow$  only mdst and udst can be processed

## Conclusion

- ▶ Belle II software publicly available ([Comput Softw Big Sci 3, 1 \(2019\)](#) and [DOI:10.5281/zenodo.5574115](#))
- ▶ C++ code with python interface
- ▶ serial execution of dynamically loaded modules to process collection of events
- ▶ Conditions Database stores settings and calibration constants
- ▶ basf2 reliable, feature-rich, fast, user-friendly, well-documented

## Conclusion

- ▶ Belle II software publicly available ([Comput Softw Big Sci 3, 1 \(2019\)](#) and [DOI:10.5281/zenodo.5574115](#))
- ▶ C++ code with python interface
- ▶ serial execution of dynamically loaded modules to process collection of events
- ▶ Conditions Database stores settings and calibration constants
- ▶ basf2 reliable, feature-rich, fast, user-friendly, well-documented

Thanks for your attention!