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

Duke UNIVERSITY

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

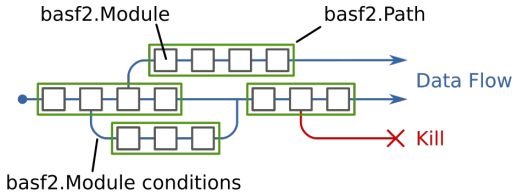
- collection of scripts to prepare environment for execution of Belle II software
- b2setup
  - setting environment variables
- b2code-create, b2code-style-check, b2code-style-fix, b2code-clean
  - creating local directory for core software development and fixing style issues
- b2install-prepare, b2install-release, 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

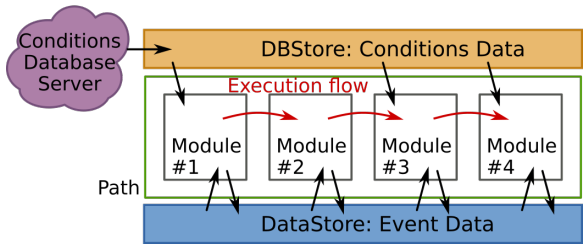
Basf2 structure

- basf2 divided into 35 packages
  - one for each subdetector: arich, cdc, ecl klm, pxd, svd, top, vxd
  - core packages: framework, reconstruction, tracking
  - data taking: daq, hlt, rawdata, trg
  - data quality: alignment, calibration, dqm, validation
  - data storage tables: mdst, skim
  - MC: decfiles, generators, geometry, simulation, structure
  - background: background, beast, ir
  - offline analysis: analysis, b2bii, mva
  - documentation / outreach: display, masterclass, online_book
- each package has one or two librarians (total number of librarians: 37)
- code written in C++
- one shared library created per package and installed in top-level lib directory
- build system based on SCons

Modular structure

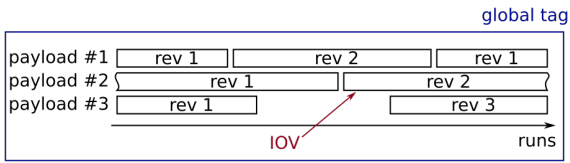
- linear arrangement of C++ modules in a path
- core functions of modules
  - initialize
  - beginRun
  - event
  - endRun
  - terminate
- python steering script to set up path





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 (iov), *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

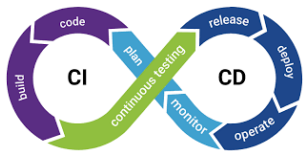
- RootInputModule
- geometry
- clustering of calorimeter
- clustering of pixel and silicon vertex detectors
- track finding
- track fitting
- track extrapolation
- track-cluster matching
- software trigger
- post-filter tracking
- PID
- RootOutputModule

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)

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



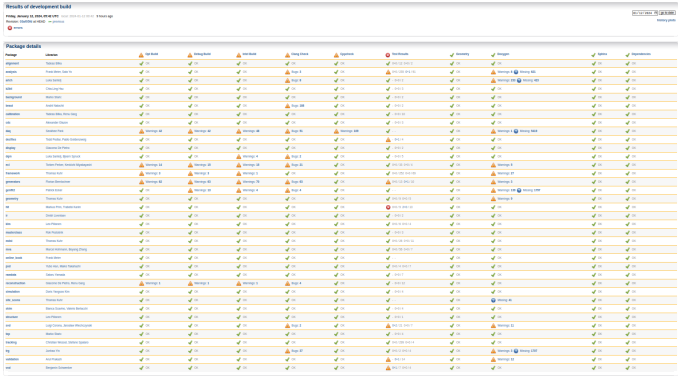
Nightly validation

- 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

Monitoring

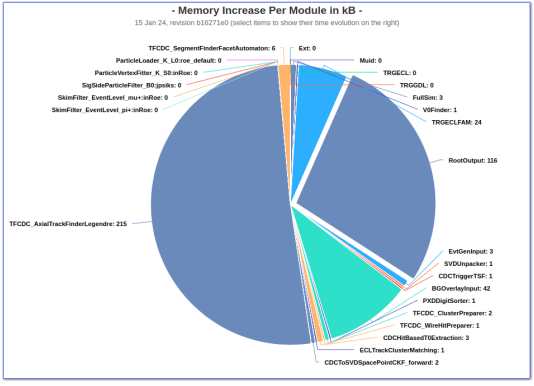
- nightly build run with different configurations (debug, intel, clang)
- many resource checks (memory consumption, execution time, output file size)
  - summarize build warnings, cppcheck, doxygen check, dependency check, geometry overlap check

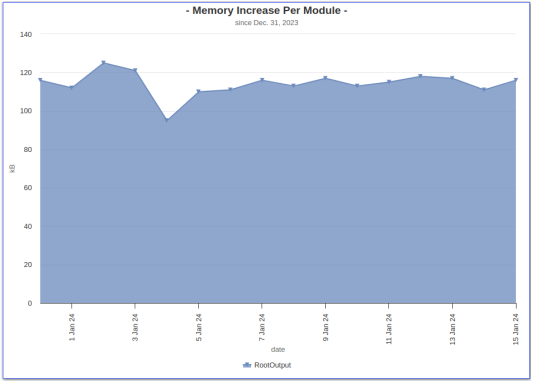


- history plots of warning and error counters as well as resource usage

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

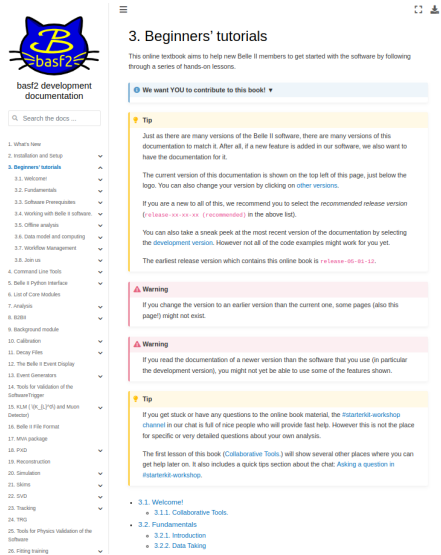




- history plots of warning and error counters as well as resource usage

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



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!