The software library of the Belle II experiment

DorisYangsoo Kim*
On behalf of the Belle II Software Group

Department of Physics, Soongsil University, 369 Sangdo-ro Dongjak-gu, Seoul 156-743, South Korea

Abstract

A next generation B factory and the detector counterpart, SuperKEKB and Belle II, are being built in Japan, as the upgrades of KEKB and Belle, respectively. The new collider will start its commissioning in 2015. This is an ambitious project. The luminosity of the e^+ e^- collider will be upgraded by a factor of 40, which will create a 50 times larger data set compared to the Belle sample. Both the background and the triggered event rates will be increased by a factor of at least 10. The Belle II software system is designed to accommodate these challenges and to run on grid, cloud, and local resources around the world. Various external software packages are employed to enhance the user interface. The software system, *basf2*, is structured as a framework built with dynamic module loading and the ability of parallel processing. The system is written in C++ with Python steering scripts, compatible with common Linux operating systems. A full detector simulation library is created based on Geant4. In this paper, we will explain the design of the Belle II software structure and the current status of the software development.

Keywords: SuperKEKB; Belle II; software; simulation; framework

1. The super B factory project

The rich physics of heavy quark decays provides creative and precise ways to look into nature. Experimentally, B factories have been producing quite prominent discoveries and new insights: The CP violation in B meson decays, the correctness of the CK picture, charm neutral meson oscillations, discovery of new particles such as X(3872) and DsJ(2370), and various other significant physics results. SuperKEKB/Belle II is an upgrade of KEKB/Belle, and an ideal environment to further exploit abundant physics topics in the field with more precision [1]. In a nutshell, the project can be summarized as follows: The luminosity of the e^+ e^- collider will be increased from $2.1 \times 10^{34}\,cm^{-2}s^{-1}$ to $8 \times 10^{35}\,cm^{-2}s^{-1}$, by a factor of 40. The size of the new data set will be 50 ab^{-1}, 50 times larger than the previous data set of 1 ab^{-1}. The detector is being upgraded to accommodate the challenges accompanying this ambitious plan.

2. The Belle II software system

basf2, the Belle II software system, is a framework system with dynamic module loading. The name is an acronym of **B**elle **A**nalysi**S** **F**ramework **2** [2]. The system is capable of parallel processing via the fork() system call by processsing different events at the same time [3]. The distribution of the software library to the worldwide collaborating GRID sites is done by CernVM-FS. The execution of jobs over the GRID

* Corresponding author. Tel: +82-2-820-0427; fax: +82-2-824-4383; e-mail: dorisykim@ssu.ac.kr.

system is done by utilizing the DIRAC INTERWARE [4-6].

The body of the software libraries is written in C++, the standard software language of high energy physics. The steering of the executable is done utilizing Python scripts [7]. The Python language can be also used to handle data sets. ROOT is used to handle input/output of the system [8].

Dedicated studies are being undertaken to find the suitable database and interface solutions. Various ideas on tracking, alignment, particle identification, and physics analysis tools have been actively pursued and tested to exploit the upgraded detector and optimize user experience.

2.1. The upgrade strategy

The basic ideas on *basf2* are the same as the previous Belle software system. However, the *basf2* code is constructed from scratch. Both systems are written in C++. The data storage format for *basf2* is ROOT I/O while that of the old one was a custom one. In addition, useful concepts are imported from the other experiments: ILC, LHCb, CDF, H1, and Alice, to incorporate modern technologies into the new library system. In addition to the external libraries already mentioned, other useful 3rd party software libraries are utilized: EvtGen, Geant4, Boost, CLHEP, etc. [9-14].

2.2. The user interface

The subversion software is employed as the revision control system [15]. The central repository of *basf2* is located at KEK. All common Linux operating systems are supported by Belle II. The main body of the code is written in C++ 11 and compiled by gcc 4.7. Clang and Intel gcc are also supported and used for cross validation.

External software packages are employed to support development of packages in a more effective way. Software developers are from all around the world. It is important to have a stable and friendly user interface. “Artistic Style” is used to format and check the style of the code [16]. Compilation and linking of *basf2* are done by SCons [17], and an automatic Buildbot system is employed to build and test the library on a daily basis during the night [18]. Fig. 1 shows the screen shots of the daily build results as an example. Documentation of *basf2* is handled by Doxygen and TWiki [19-20]. Redmine is used to track important issues and problems by software handlers [21].

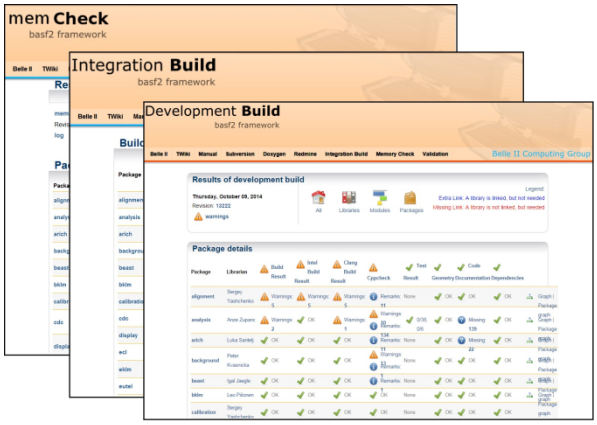

Fig. 1. The screen shots of daily build results. The color scheme of the interface is slightly changed from the original images for better print quality.

2.3. The basic structure of basf2

The basic processing unit of *basf2* is a module. Its role could be a simple one such as reading data from a file or complex ones such as simulation or tracking. All work is done in modules, which are usually written in C++. A user can decide what kind of modules would be executed and in what order that should be done [2].

Usually, processing of an event is represented by a linear chain of modules on a path. The creation of the path, loading of modules, and executing of the path are defined as commands in a Python script.

Depending on the return code of a module, execution of an event continues in one or the other path. This strategy would be useful for creating skimmed data sets.

During the processing of the event, the data sets needed by *basf2* are stored in “DataStore”, a common storage area for data. All data exchanges between modules are executed through the Datastore [2].

3. Detector geometry

The geometry parameter values describing the Belle II detector are stored in a central repository in the XML format. For simulation, the actual geometry is created by C++ algorithms based on the repository parameters. For reconstruction, the geometry is converted to the ROOT TGeo format using the VGM library [22]. The parameters needed for digitization are imported directly from the XML repository [1].

Since the XML format is a markup language, the parameter values are directly available to users.

Modification, maintenance, and updates of the values are easy and can be done quickly.

4. An example Python script

In *basf2*, the full cycle of an event is defined as a chain of modules representing event generation, geometry creation, simulation, reconstruction, and user analysis. In future, when real data is generated from beam collisions at SuperKEKB, the simulation part would be replaced by modules handling raw data. The following paragraph is an example python script needed to run a full cycle *basf2* job:

```
# ---   A basf2 steering script   ---
from basf2 import *
from simulation import add_simulation
from reconstruction import  add_reconstruction
main = create_path()

# meta information (100 events in Run 1)
event_info = register_module ('EventInfoSetter')
event_info.param('evtNum', [100])
event_info.param('run', [1])
main.add_module(event_info)

# event generation
generator = register_module('EvtGen')
main.add_module(generator)

# simulation
add_simulation(main)

# reconstruction
add_reconstruction(main)

# user analysis
…

process(main)

# ---   End of the job   ---
```

5. Simulation

External generator libraries are employed to provide production and decay of particles created by e^+ e^- collisions and background processes. For example, EvtGen is used to create B pair events. Geant4 is used to undertake full detector simulation. Digitization of hits created in the sensitive detector area is a separate package written by the corresponding sub-detector group [1].

Due to high luminosity of SuperKEKB beams, more than one physics interaction can happen in one event. One interaction would be the desired physics process while the others would be background events. Special care has been taken to mimic this phenomenon in the *basf2* simulation. First, background events are simulated using Geant4. Several background generators are used including Touschek and Bhabha processes. The detector hits created in this process are stored as a separate data set. Next, physics events are simulated and hits are created in the sensitive detector area. Then the hits from two contributions are merged and sent to the digitization step together. This procedure is similar to what happens in reality [1].

6. Tools in development

To provide reliable methods and to reduce costly resources needed for complicated physics analysis, various software tools are being developed in parallel as a part of *basf2*. Users can utilize these tools by loading the corresponding modules in their *basf2* scripts.

6.1. Tracking and alignment tools

Reconstruction of charged tracks proceeds separately in the central drift chamber and in the vertex detector. For the drift chamber, two algorithms are used: A global track finder based on Legendre transformations, and a local finder based on cellular automata. In addition, an old finder algorithm used in the previous Belle central drift chamber has been ported as a reference. In the vertex detector, the finder is a combination of two ideas: A cellular automaton for identifying track candidates and a Hopfield network to resolve overlapping sets of candidates [23]. Tracks crossing both detectors are combined using geometrical matching. For track fitting, a detector independent library is employed [24], which is a rewrite of GENFIT [25]. Another aim of this rewrite was to provide support for alignment and calibration. The default algorithm used in fitting is the "deterministic annealing filter" [26], which handles wrongly assigned (background) hits gracefully.

Alignment and calibration of tracking detectors are being developed using the Millepede II package [27], which is interfaced through the track fitting library and in particular, via its implementation of the "general broken lines" fitting method [28]. For the verification

of the implementation procedure, the test beam data sets are used [29].

6.2. Particle identification tools

Particle identification (PID) tools are being created in two groups: The neutral particles and the charged particles. The particles handled in the neutral group are π^0, photon, K_L and K_S. In the charged group, the selection tools for e, μ, $\pi^{\pm}$, $K^{\pm}$, protons are included. Feasibility for deuteron reconstruction is also being studied in this group. For each track candidate, particle identification information is stored in an object called "PidLikelihood". The log likelihoods from all the sub-detectors are saved in the object and an interface is created for various identification hypotheses. For example, the identification information collected from the time of propagation and the ring imaging Cherenkov detectors and the *dE/dx* information obtained from tracking detectors is merged together to create a likelihood hypothesis function, which can be used to separate $\pi^{\pm}$ and $K^{\pm}$. Another example is the μ identification, which is done by extrapolating a charged track with the muon detector hits via the Kalman filter.

6.3. Physics analysis tools

Common analysis tools are being developed to speed up the analysis process and enhance quality control. One of the first analysis tools developed is "Particle'' class, which is a common representation of all particle types: Final state particles, pre-reconstructed ones such as K_S and Lambda, and those ones reconstructed from their decays. Then the modular analysis tools are applied to the candidates stored in the "ParticleList".

Basic analysis tools are dealing with particle reconstruction and Monte Carlo truth matching. More advanced tools are in development to reconstruct B decay particles with excellent efficiency and low background rate, based on the updated detector performance: A tool to suppress continuum background events, a vertex reconstruction tool to tag B particles, flavour tagging tools, etc. In addition, methods needed to reconstruct the full event topology as well as many other ideas are being tested to provide an excellent analysis environment [30]. Fig. 2 shows an example of the dependencies between decay channels/particles for full reconstruction of the $D^{*+} \rightarrow D^0 \pi^+$ event.

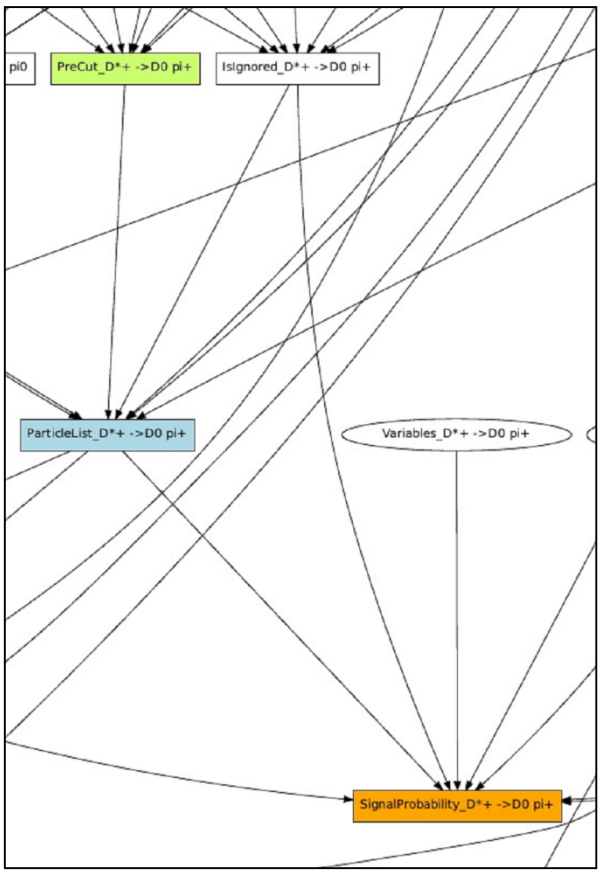

Fig. 2. An example of dependencies between decay channels/particles for full reconstruction of the D*+ →D0 π+ event.

7. Utility packages

Since the Belle II software system is in the development stage, *basf2* is evolving constantly with improvements and new components. To check the accuracy of modifications and stability of the library, a utility package for validation is created, which displays basic histograms generated by *basf2*. Several data sets are overlayed in each histogram. The data sets consist of reference samples and most recent library versions. Parallel execution of validation job scripts is available, which speeds up the process. The histograms are updated every day and shown in the Belle II website, so software shifters can catch any problems stemming from changes in the library. Fig. 3 shows a screen shot of the validation histogram display. In addition, we employ the frameworks for unit tests based on Google Test and for run tests [31].

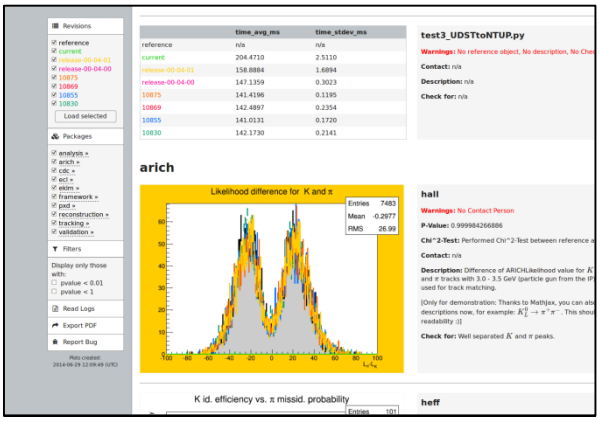

Fig. 3. A screenshot of the histograms created by the validation package. The left column shows display options and command buttons: The samples used for histogram overlays, the available packages and tools, filters, buttons to export the results to a PDF file and report a new Redmine issue.

Another useful tool is the event display package, which is invoked by *basf2* with support from ROOT and OpenGL [32].

8. Summary

The Belle II experiment is a collaboration of over 600 scientists from 96 institutions in 26 countries. The collaboration is actively engaging in development of a software system suitable for the next generation B factory. Various ideas have been tested and successful ones are incorporated as library packages. A full cycle of *basf2* run from event generation to user analysis has been successfully conducted. Large scale Monte Carlo campaigns are undertaken regularly, testing the software system and creating the challenge data sets.

The amount of C++ code included in the current version of *basf2* is about 280k lines, excluding comments and white space. The library is developing at a steady pace and is expected to be ready when SuperKEKB starts physics runs in 2017.

Acknowledgments

This work was supported in part by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (2013R1A1A3007772) and by the Supercomputing Center/Korea Institute of Science and Technology Information with supercomputing resources including technical support (KSC-2012-C1-19).

References

[1] Edited by Z. Dolezal and S. Uno, "Belle II technical design report," *KEK Report* 2010-1; arXiv:1011.0352, Oct., 2010.
[2] A. Moll, Proc. International Conference on Computing in High Energy and Nuclear Physics 2010 (CHEP2010), Academia Sinica, Taipei, Taiwan, 18-22 Oct., 2010; J. Phys.: Conf. Ser., 331 (2011) 032024.
[3] R. Itoh et al., Proc. International Conference on Computing in High Energy and Nuclear Physics 2012 (CHEP2012), New York, USA, 21-25 May, 2012; J. Phys.: Conf. Ser., 396 (2012) 022026.
[4] T. Kuhr, Proc. 20th International Conference on Computing in High Energy and Nuclear Physics 2013 (CHEP2013), Ambsterdam, The Netherlands, 14-18 Oct., 2013; J. Phys.: Conf. Ser., 513 (2014) 032050.
[5] P. Buncic et al., Proc. 12th International Workshop on Advanced Computing and Analysis Techniques in Physics Resaerch (ACAT08), Erice, Italy, 3-7 Nov., 2008 (ACAT08); Proc. Sci. (ACAT08) 012.
[6] DIRAC INTERWARE, http://diracgrid.org.
[7] Python, http://www.python.org.
[8] ROOT, http://root.cern.ch.
[9] EvtGen, http://evtgen.warwick.ac.uk.
[10] A. Ryd and D. Lange, "The EvtGen event generator package", Proc. International Conference on Computing in High Energy Physics and Nuclear Physics 1998, Chicago, IL. USA, Aug. 31 – Sep. 4, 1998.
[11] S. Agostinelli et al., Nucl. Instr. Meth. Phys. Res. A 506 (2003) 250
[12] J. Allison et al., IEEE T. Nucl. Sci. 53, no. 1, (2006) 270.
[13] Boost, http://www.boost.org.
[14] CLHEP, http://proj-clhep.web.cern.ch.
[15] Apache Subversion, http://subversion.apache.org.
[16] Artistic Style, http://astyle.sourceforge.net.
[17] SCons, http://www.scons.org.
[18] Buildbot, http://www.buildbot.net.
[19] Doxygen, http://www.doxygen.org.
[20] TWiki, http://twiki.org.
[21] Redmine, http://www.redmine.org.
[22] I. Hřivnáčová, Proc. 14th International Conference on Computing in High Energy and Nuclear Physics 2004 (CHEP 2004), Interlaken, Switzerland, 27 Sep.-1 Oct., 2004; CERN-2005-002 (2005) 345.
[23] J. Lettenbichler et al., Proc. International Conference on Computing in High Energy and Nuclear Physics 2012 (CHEP2012), New York, USA, 21-25 May, 2012; J. Phys.: Conf. Ser., 396 (2012) 022030.
[24] J. Rauch and T. Schlüter, "GENFIT – a generic track fitting toolkit", submitted for publication, Proc. 16th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT2014), Prague, Czech Republic, 1-5 Sep., 2014; To appear in IOP Conference Series; arXiv:1410.3608, Oct., 2014.
[25] C. Höppner et al., Nucl. Instr. Meth. Phys. Res. A 620 (2010) 518.
[26] R. Frühwirth and A. Strandlie, Nucl. Instr. Meth. Phys. Res. A 559 (2006) 162.
[27] V. Blobel, Nucl. Instr. Meth. Phys. Res. A 566 (2006) 5.
[28] C. Kleinwort, Nucl. Instr. Meth. Phys. Res. A 673 (2012) 107.
[29] T. Bilka, G. Casarosa, R. Frühwirth, C. Kleinwort, P. Kodys et al., "Demonstrator of the Belle II online tracking and pixel data

reduction on the high level trigger system", submitted to Trans. Nucl. Sci.; Proc. 19th Real Time Conference (RT2014), Nara, Japan, 26-30 May, 2014; arXiv:1406.4955, Jun., 2014.
[30] C. Pulvermacher, T. Ceck, M. Feindt, M. Heck, and T. Kuhr, "An automated framework for hierarchical reconstruction of B mesons at the Belle II experiment", submitted for publication, Proc. 16th International Workshop on Advanced Computing and Analysis Techniques in Physics Research (ACAT2014), Prague, Czech Republic, 1-5 Sep., 2014; To appear in IOP Conference Series; arXiv:1410.3259, Oct., 2014.
[31] Google Test, http://code.google.com/p/googletest.
[32] OpenGL, https://www.opengl.org.