EPJ Web of Conferences **214**, 09004 (2019) https://doi.org/10.1051/epjconf/201921409004
CHEP 2018

Belle II at the Start of Data Taking

Thomas Kuhr[1,*] for the Belle II Software Group

[1]Ludwig-Maximilians-Universität München, Excellence Cluster Universe, Boltzmannstr. 2, 85748 Garching, Germany

Abstract. The Belle II experiment is expected to collect e^+e^- collision data at a 40 times higher instantaneous luminosity than achieved so far. The high collision rate requires not only an upgrade of the detector, but also of the computing system and the software to handle the data. The first collision data taken during the commissioning run in Spring 2018 provides an excellent opportunity to review the status of the Belle II computing and software and assess its readiness for the physics data taking starting 2019.

1 Introduction

The search for new physics beyond the standard model is one of the main tasks of experimental particle physics. A complementary approach to searches for the production of new particles in high-energy collisions are indirect searches that look for contributions of virtual new particles to low energy processes. Huge numbers of events have to be collected to be sensitive to tiny deviations from the Standard Model predictions induced by new particles. An example, where the current combination of measurements from BaBar, Belle, and LHCb shows a deviation of more than 3 standard deviations are the ratios $R_{D^{(*)}} = \mathcal{B}(B \to D^{(*)}\tau\nu)/\mathcal{B}(B \to D^{(*)}\ell\nu)$ [1].

Much more data is needed to confirm or disprove this anomaly or to find other hints. The KEKB e^+e^- accelerator is upgraded to the SuperKEKB accelerator to reach an instantaneous luminosity 40 times higher. The higher event and background rates also require an upgrade of the detector [2]. The Belle II detector is designed for precision measurements of B meson, charm hadron, and tau lepton decays.

After a commissioning run of SuperKEKB without final focusing magnets in 2016 (phase 1) the first e^+e^- collisions where recorded by the Belle II detector on April 26th, 2018. During the 2018 commissioning run, called phase 2, the inner part of the detector was equipped with devices for the measurement of beam backgrounds instead of the final vertex detector. The start of the physics run with a vertex detector is scheduled for early 2019.

Computing and software were key ingredients to the successful start of the Belle II data taking. The event display provided immediate visual proof of first collisions and the readiness of the processing chains and software algorithms allowed to present the first performance plots to the public in only two weeks.

*e-mail: Thomas.Kuhr@lmu.de

2 Data Flow and Computing Model

At design luminosity the Belle II detector is expected to deliver a raw data rate of 2 GB/s. The raw data is processed at KEK and copied to remote sites, initially only to BNL, then staring in 2021 to several raw data centers in America, Europe, and Asia. The output of the reconstruction are tracks, clusters, and particle identification information stored in the mini data summary table (mDST) format, which is about an order of magnitude smaller than the raw data. This format is also the output of the simulation. The mDST contains all information required for physics analysis. To reduce the number of events that has to be processed during analysis, skims containing only events fulfilling certain selection requirements are produced. The selection is based on particle candidates reconstructed from the mDST information. These particle objects are retained in the skimmed data, then called uDST, to avoid their recreation in each analysis. The skims are usually the input of user analysis jobs which produce ntuple data output for analysis on local resources.

As Belle II is a very international collaboration with more than 800 members from about 100 institutions in 25 different countries a distributed computing model similar to that of the LHC experiments is adopted. The above mentioned raw data centers are responsible for the reprocessing of their share of the raw data. In addition, regional data centers in several countries store mDST and uDST data. Simulated data is produced at raw and regional data centers as well as at MC production sites that have no or very limited storage capacities. The latter can be classical grid sites, or sites using cloud technologies, or local clusters.

The distributed computing system is based on DIRAC [3] and exploits some other existing tools: LFC and AMGA [4] are used as file and metadata catalog. File transfers are managed by FTS and the software is distributed with CVMFS [5]. The scalability of the distributed computing system was tested in several Monte-Carlo production campaigns, see Figure 1. A peak usage of 25k concurrent jobs was reached.

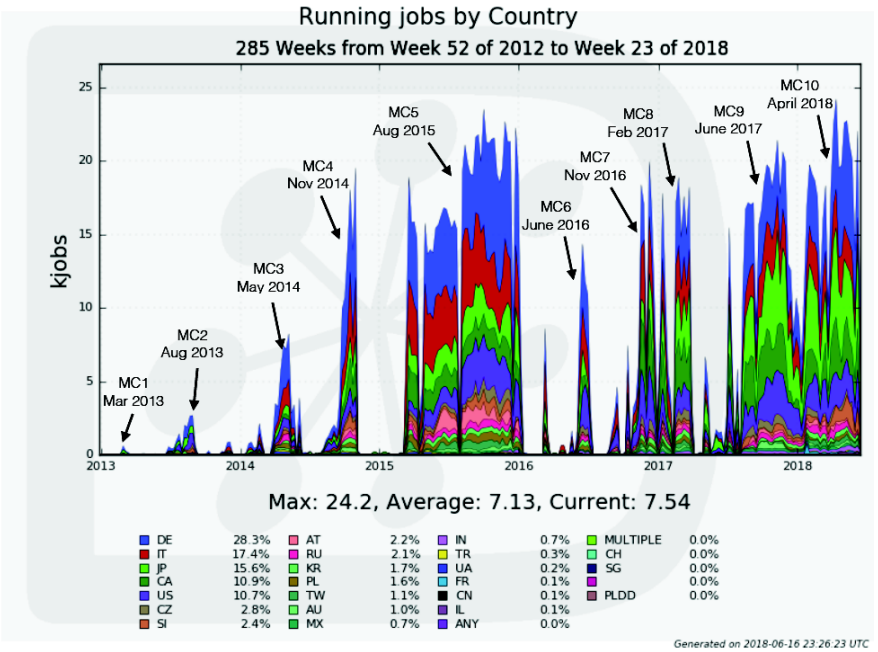

Figure 1. Number of running jobs in MC production campaigns since 2013

3 Software Framework

The Belle II Analysis Software Framework, basf2, is used online for the data acquisition and offline for the simulation, reconstruction, display, and analysis of data. A detailed description can be found in Ref. [6]. The framework supports dynamic loading of modules to flexibly construct a chain of algorithms, called path, for the processing of events according to the task at hand. The event data is shared between modules via a global interface, the DataStore. The DataStore handles also relations between event data objects, allowing to dynamically manage connections between objects without having to specify them in the class definition. Figure 2 illustrates the DataStore interface.

```
StoreArray<Track> tracks;
for (const Track* track:tracks) {
  const PIDLikelihood* pid = track->getRelated<PIDLikelihood>();
}
```

Figure 2. Loop over tracks and their related PID information in the DataStore.

The ROOT format [7] is used for persistent storage of data. An interface to read mDST data of the Belle experiment is implemented. This enabled analyses of real data well before the start of the Belle II data taking and provided valuable feedback for the software development.

The configuration of modules and paths is done in Python, providing a powerful way for creating meta-frameworks. As the framework is aware of the dependencies between data objects and modules, this information can be visualized, as shown in Figure 3, and is included in an automatically generated module documentation.

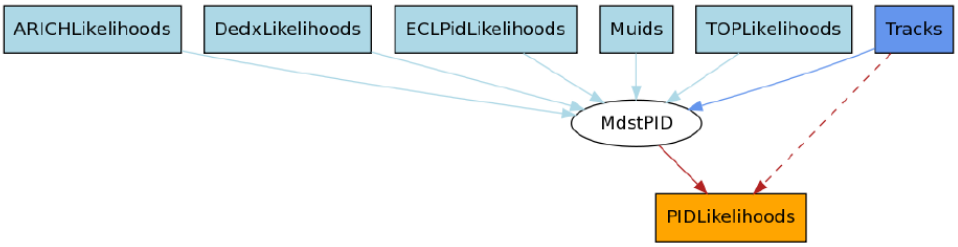

Figure 3. Dependencies between data objects (rectangles) and modules (ellipses). Relations are shown as dashed lines.

As the Belle II event data is small compared to that of the LHC experiments the processing of multiple events in parallel is no problem in terms of memory usage. The framework supports this using forked processes. We observe good scaling of the execution performance with the number of parallel processes up to the number of available cores. Memory is saved in parallel processing mode because the detector geometry is created before the forking and thus stored in memory only once.

The framework provides transparent access to conditions data with an interface that is very similar to that for event data. The conditions are stored as objects in ROOT files, called payloads. They are managed by a conditions database service and distributed via CVMFS.

The server hosts a database that keeps the assignments of intervals of validity, expressed in run ranges, to payloads [8].

4 Simulation

Several event generators are available within basf2 to support the generation of signal and background events for all physics analyses performed by Belle II collaborators. GEANT4 [11] is used to simulate the energy depositions in the sensitive volumes, called SimHits, of the particles traversing the detector. As shown in Figure 4, the most time consuming part is the simulation of optical photons in the barrel particle identification detector (TOP). The SimHits are the input to digitizers implemented as detector-specific modules. The output digits can be converted to the format of raw data and back using so-called packer and unpacker modules. While the simulation and its parts can be run in separate jobs it is usually executed in a single job together with the reconstruction and only the mDST output is retained.

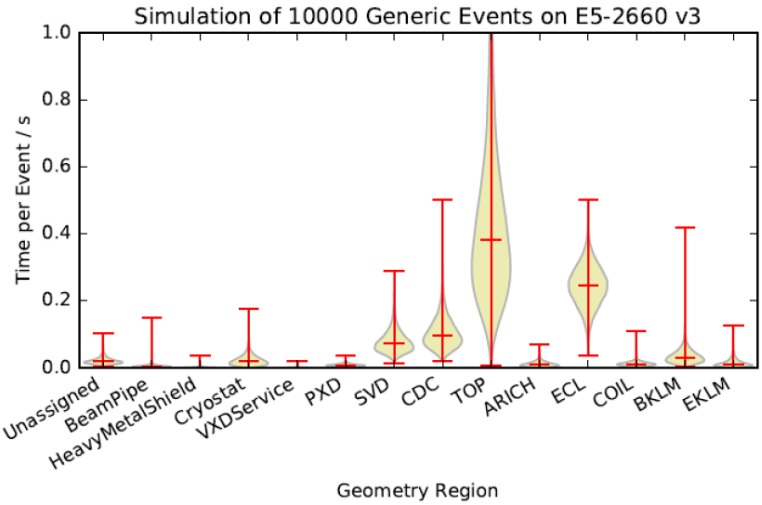

Figure 4. Distributions of time spent in the GEANT4 simulation by detector component.

Two methods have been developed to include the effect of beam background in the simulation. The first method, called background mixing, relies on simulated background. The SimHits of the simulated background are combined with those from simulated signal before digitization. The second method, called background overlay, combines digits in a detector-specific way. While this may lead to approximations in case signal and background contribute to the same channel, it allows to use real data taken with a random trigger for the background.

For education and outreach purposes simulated events can be displayed in a virtual reality environment using head mounted displays or shutter glasses.

5 Reconstruction

An essential part of the event reconstruction is the tracking [9, 10]. One challenge is the low transverse momentum of tracks, typically a few hundred MeV/c. For a complete event

reconstruction a good reconstruction efficiency even at 50 MeV/c is desirable. Multiple scattering becomes a concern for track finding and fitting. Another challenge is the high density of (background) hits in the innermost detector layer, being about two orders of magnitude higher than at CMS.

A Legendre and a cellular automaton algorithm are used to find tracks in the Central Drift Chamber (CDC). An independent track finding algorithm in the Silicon Vertex Detector (SVD), consisting of 4 layers of double-sided strip sensors, is complemented by an extrapolation of CDC tracks to the SVD using a combinatorial Kalman filter (CKF). A CKF is also used to attach hits in the two innermost detector layers comprising the Pixel Vertex Detector (PXD). Track fitting is done with GENFIT2 [12] using a deterministic annealing filter (DAF). The track reconstruction strategy is summarized in Figure 5.

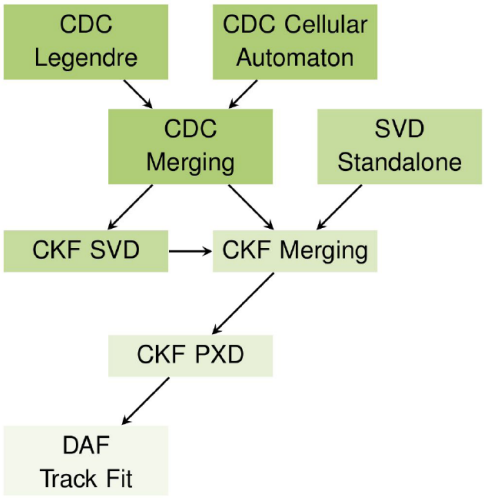

Figure 5. Track reconstruction strategy.

Background is also an issue for the reconstruction of the energy and position of clusters in the electromagnetic calorimeter (ECL). New algorithms using hypothesis-specific optimizations are implemented. The identification of charged particles (PID) relies on likelihoods for the electron, muon, pion, kaon, proton, and deuteron hypotheses determined for each detector component that provides discriminating information. The likelihoods are independent of the analysis-specific selection and can be easily combined in a sum of log-likelihoods over detectors to obtain optimal PID information. Analysis-dependent prior probabilities still have to be implemented.

A Calibration Framework (CAF) based on AirFlow [13], a platfrom for scheduling dependent tasks, was implemented to provide a common infrastructure for alignment and calibration algorithms [14]. One example of such an algorithm is the simultaneous alignment of multiple detectors with Millepede II [15]. It was executed on first data and an improved track parameter resolution was clearly visible with each further iteration of the data processing chain.

6 Analysis

Several modules for common analysis tasks are available. This includes the reconstruction of particles from tracks and clusters, the combination of particles, the selection of particles, the matching of MC truth information, vertex fitting [16], flavor tagging [17], suppression of events without B mesons, and a full event interpretation [18]. These analysis building blocks can be easily configured and combined using Python functions. Figure 6 shows an example. The framework can also be used with a Jupyter notebook which is, for example, used very successfully in the Belle II StarterKit for the education of new collaborators.

```
# create Ks -> pi+ pi- list from V0
# keep only candidates with 0.4 < M(pipi) < 0.6 GeV
fillParticleList('K_S0:pipi -> pi+ pi-', '0.4 < M < 0.6')

# reconstruct J/psi -> mu+ mu- decay
# keep only candidates with 3.0 < M(mumu) < 3.2 GeV
reconstructDecay('J/psi:mumu -> mu+:loose mu-:loose', '3.0 < M < 3.2')

# reconstruct B0 -> J/psi Ks decay
# keep only candidates with 5.2 < M(J/PsiKs) < 5.4 GeV
reconstructDecay('B0:jspiks -> J/psi:mumu K_S0:pipi', '5.2 < M < 5.4')

# perform B0 kinematic vertex fit using only the mu+ mu-
# keep candidates only passing C.L. value of the fit > 0.0 (no cut)
vertexRave('B0:jspiks', 0.0, 'B0 -> [J/psi -> ^mu+ ^mu-] K_S0')

# build the rest of the event associated to the B0
buildRestOfEvent('B0:jspiks')

# perform MC matching (MC truth association). Always before TagV
matchMCTruth('B0:jspiks')

# calculate the Tag Vertex and Delta t (in ps)
# breco: type of MC association.
TagV('B0:jspiks', 'breco')
```

Figure 6. Example of Python code snippet to reconstruct the decay $B^0 \to J/\psi(\to \mu^+\mu^-)K^0_S(\to \pi^+\pi^-)$, check whether it matches the MC truth, fit the vertex of the B^0 and of the remaining tracks.

With the full software stack, including high-level analysis tools, and the resources and procedures for the data processing being ready when first data arrived it was easy for all collaborators to analyze it. As a result, the first performance plots like the ones in Figure 7 could be presented to the public only two weeks after the first collisions. With more data also charm and B mesons are seen.

7 Software Development

What was the software development process behind this achievement?

The Belle II software is composed of three parts: scripts for the installation and environment setup, the external software, and the Belle II specific code. The latter is written in C++17 and Python3 and organized in top-level directories, called packages, with assigned responsible librarians. The Atlassian tools, provided by DESY, are used for the code management. The Bitbucket server is configured to allow direct pushes to the master only for

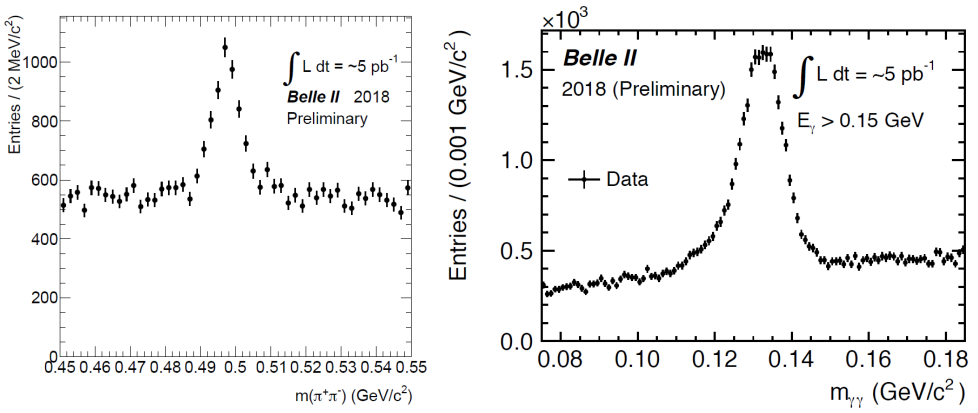

Figure 7. Reconstructed K_S^0 and π^0 candidates using first Belle II data.

librarians and people they trust. Every Belle II member is allowed to create feature branches and make pull requests.

Typically two major versions of the software are released per year. Light versions containing a subset of packages needed for analysis are released more frequently. In addition monthly builds are performed as a periodic motivation for the librarians to review the changes in their package.

The build system based on SCons [19] is set up in a way that developers usually do not have to care about it. The only required build configuration is to specify the list of linked libraries. Every other build configuration is done by directory naming convention.

Software quality is checked at various levels. Hooks for commits to the local or central repository check the code style and the file size. Unit and script tests can be executed by the developer with a simple command and are run on commits to the central repository. The Bamboo [20] and the Buildbot [21] continuous integration systems are used for this. The latter also performs a more extensive nightly check, e.g. including a check for memory leaks, and runs a set of validation jobs. They produce plots that are then compared with previous versions and a reference on a web page. A software quality shifter watches for regressions. Other monitored quantities are the execution time, memory usage, and output size of a standard job. A thorough validation by physicists that cannot be fully automated is done for each major release.

Documentation for developers is automatically generated from the source code using Doxygen [22]. Sphinx [23] is adopted to provide comprehensive documentation for users. Questions and answers are managed with an askbot instance [24].

8 Outlook and Conclusions

While software and computing significantly contributed to the success of the first Belle II data taking, some areas with potential for improvement have been identified. Documentation has significantly improved, but some parts are still missing. The complex task of conditions management is not yet solved satisfactorily. Release validation is a major effort that could probably be streamlined further. The Belle II code is not public and has no license. The integration with the online and distributed computing systems could be improved. Progress has recently been made regarding publications of achievements in software development.

Key factors for success are skilled and motivated people and good communication. Developers must identify with the project and care about its success. This is achieved by an open and transparent development process, a low threshold for contributions, and the sharing of responsibilities. On the other hand one has to be careful that developers do not get too attached to their own code, but see it as a contribution to a collaborative effort. The pull request workflow helps to achieve this because all code is reviewed and potentially improved by somebody else than the author. Clear and commonly accepted rules and procedures also strengthen the perception of work as a collaborative effort.

Several measures are taken to make it easy to contribute. The software can be installed easily with a few commands, has minimal system requirements and works on various systems so that developers can work on their system of choice. No condition other than being a Belle II member has to be met to be allowed to make pull requests. Although some rules, like the code style, are enforced, following them is usually easy because for example a style formatting tool is provided. In general, feedback on code issue is given as quickly as possible which reduces the effort to fix them.

We pay attention to code quality, resource usage, documentation, and validation. The continuous integration with automated tests is an essential tool for this. Further means to improve quality are pull request reviews and release validation.

Finally, our goal is to make the software easy to use. Effective user communication channels are important. And taking the users perspective helps to identify problems and to set priorities.

In summary, the Belle II software, the computing system, and the production procedures were in place to successfully provide the first collision data for analyses. The valuable feedback obtained helps to make further progress in the transition from work with simulation only to the case of real data with realistic imperfections. The main challenge is to keep the skilled and motivated team of software developers and supporters so that we will be able to exploit the full, exciting physics potential of 50 ab^{-1} of Belle II data.

References

[1] Heavy Flavor Averaging Group (HFLAV), https://hflav-eos.web.cern.ch/hflav-eos/semi/summer18/RDRDs.html (2018)
[2] T. Abe *et al.* (Belle II Collaboration), arXiv:1011.0352 (2010)
[3] A. Tsaregorodtsev1 *et al.*, J. Phys. Conf. Ser. **119**, 062048 (2008)
[4] N. Santos and B. Koblitz, Nucl. Instrum. Meth. A **559**, 53 (2006)
[5] https://cernvm.cern.ch/portal/filesystem (2018)
[6] T. Kuhr *et al.* (Belle II Framework Software Group), arXiv:1809.04299 (2018)
[7] R. Brun and F. Rademakers, Nucl. Instrum. Meth. A **389**, 81 (1997)
[8] L. Wood *et al.*, "Performance of the Belle II Conditions Database" in these proceedings (2018)
[9] T. Hauth for the Belle II Tracking Group, "Belle II Track Reconstruction and Results from first Collisions" in these proceedings (2018)
[10] S. Spataro for the Belle II Tracking Group, "Track Fitting for the Belle II Experiment" in these proceedings (2018)
[11] S. Agostinelli *et al.* (GEANT4 Collaboration), Nucl. Instrum. Meth. A **506**, 250 (2003)
[12] J. Rauch and T. Schlüter, J. Phys. Conf. Ser. **608**, 012042 (2015)
[13] https://airflow.apache.org (2018)
[14] T. Bilka *et al.*, "Alignment and Calibration of the Belle II Detector" in these proceedings (2018)

[15] V. Blobel and C. Kleinwort, Proceedings of the Conference on Adcanced Statistical Techniques in Particle Physics (2002)
[16] F. Tenchini and J.-F. Krohn for the Belle II Software Group "Decay Chain Reconstruction at Belle II" in these proceedings (2018)
[17] F. Abudinen, "The Belle II flavor tagger" in these proceedings (2018)
[18] T. Keck *et al.*, arXiv:1807.08680 (2018)
[19] http://www.scons.org (2018)
[20] https://www.atlassian.com/software/bamboo (2018)
[21] http://buildbot.net (2018)
[22] http://www.doxygen.nl (2018)
[23] http://www.sphinx-doc.org (2018)
[24] https://askbot.com (2018)