Software and Physics Simulation at Belle II

Doris Yangsoo Kim
Soongsil University

On behalf of the Belle II Collaboration

August 6, 2015

DPF 2015 Ann Arbor, Michigan

The Super B Factory Project

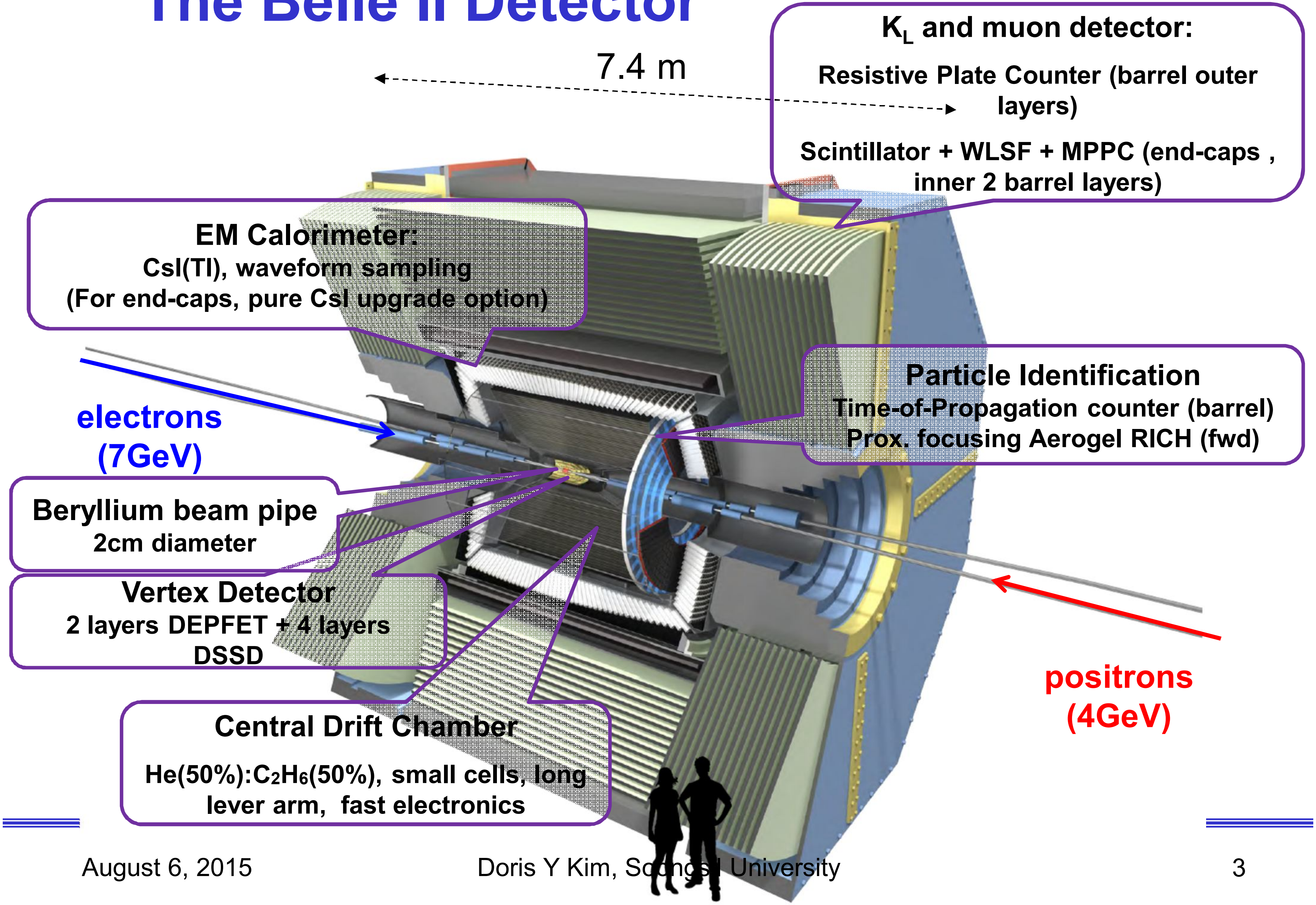
Upgrade of KEKB/Belle

	KEKB		SuperKEKB	
Luminosity:	$2.1\text{x}10^{34}\ \text{s}^{-1}\text{cm}^{-2}$	→	$8\text{x}10^{35}\ \text{s}^{-1}\text{cm}^{-2}$	(x 40)
Total Data:	$1\ \text{ab}^{-1}$	→	$50\ \text{ab}^{-1}$	(x 50)
Detector:	Belle	→	Belle II	

For details, please refer to Prof. Carl Rosenfeld's talk on August 5, 5:00 pm in Accelerators, Detectors, Computing

"Status of Belle II and SuperKEKB"

The Belle II Detector



K_L and muon detector:

Resistive Plate Counter (barrel outer layers)

Scintillator + WLSF + MPPC (end-caps , inner 2 barrel layers)

EM Calorimeter:
CsI(Tl), waveform sampling
(For end-caps, pure CsI upgrade option)

Particle Identification
Time-of-Propagation counter (barrel)
Prox. focusing Aerogel RICH (fwd)

Beryllium beam pipe
2cm diameter

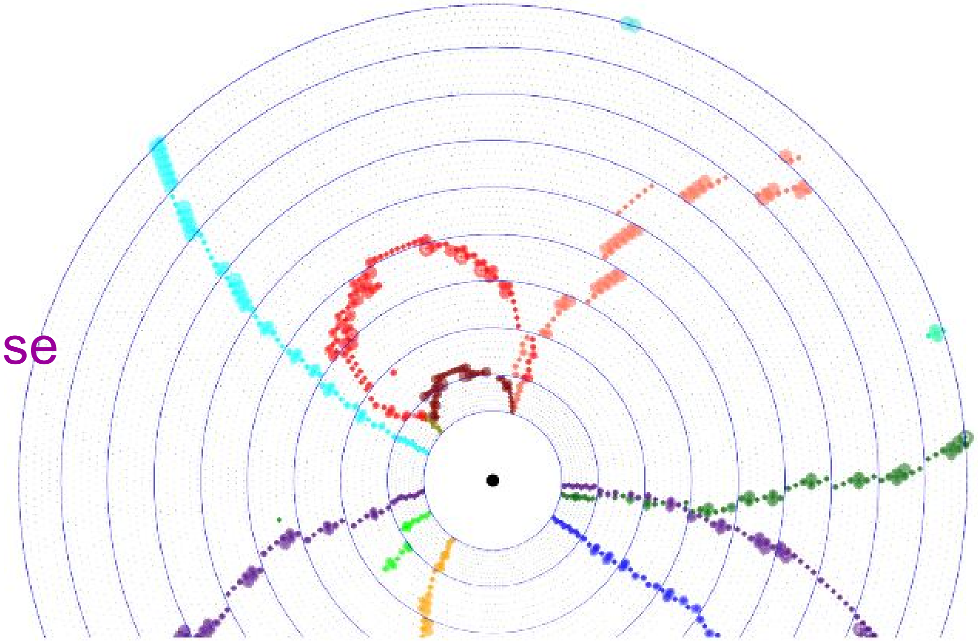
Vertex Detector
2 layers DEPFET + 4 layers DSSD

Central Drift Chamber

He(50%):C_2H_6(50%), small cells, long lever arm, fast electronics

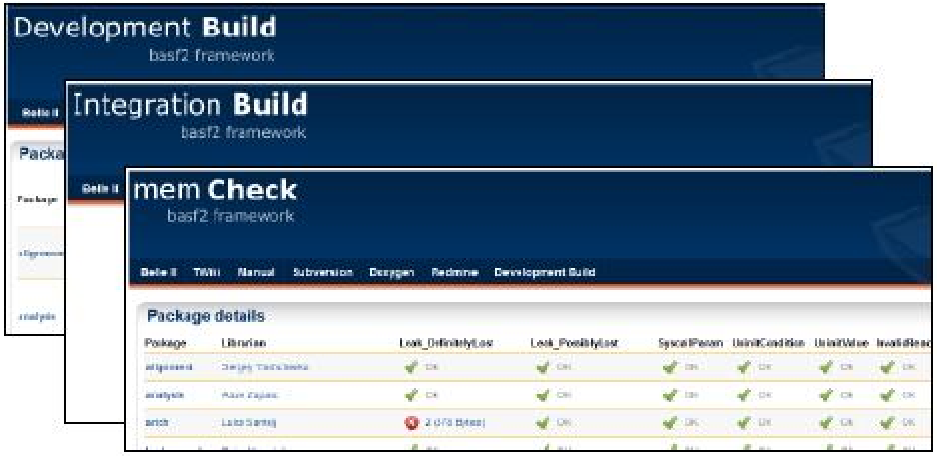
The Belle II Software System

- A “framework” system with dynamic module loading, parallel processing, Python steering, and ROOT I/O
- Use of GRID with Dirac (Dr. Vikas Bansal’s talk next)
- Full detector simulation with Geant4.9.6.2 (Plan to upgrade to v10)
- Tracking with Cellular Automaton and Legendre finder
- Calibration and Alignment
  - Millepede II, GBL
  - Testbeam data utilized.
- Distributed Conditions Database
- PID tools
- Analysis tools



User Interface

- Central code management systems at KEK: The Subversion software
- All common linux operating systems supported: SL, Ubuntu, etc
- C++ 11 and gcc 4.9.2, clang 3.4, icc14
- Formatting tool: astyle for C++, pep8/autopep8 for python
- Building: scons and buildbot system
  - Daily automatic building and validation for regression tests.
  - Monthly integration build
  - Release with features
- Googletest, valgrind, cppcheck
- Documentation
  - Doxygen, Twiki
- Issue tracking: Redmine



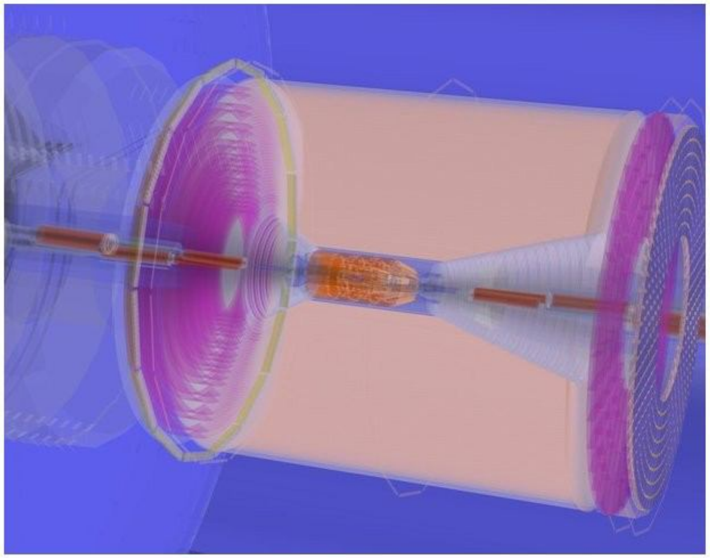
Package details

Package	Librarian	Leak_DefinitelyLost	Leak_PossiblyLost	SyscallParam	UninitCondition	UninitValue	InvalidReac
alignment	Sergey Yashchenko	✓ OK	✓ OK	✓ OK	✓ OK	✓ OK	✓ OK
analysis	Anze Zupanc	✓ OK	✓ OK	✓ OK	✓ OK	✓ OK	✓ OK
arich	Luka Santelj	✗ 2 ([illegible] Bytes)	✓ OK	✓ OK	✓ OK	✓ OK	✓ OK

The Upgrade of the Software from Belle

- BASF2 (**B**elle **A**nalysi**S** **F**ramework 2)
  Basic ideas from the Belle software system + Constructed from scratch.

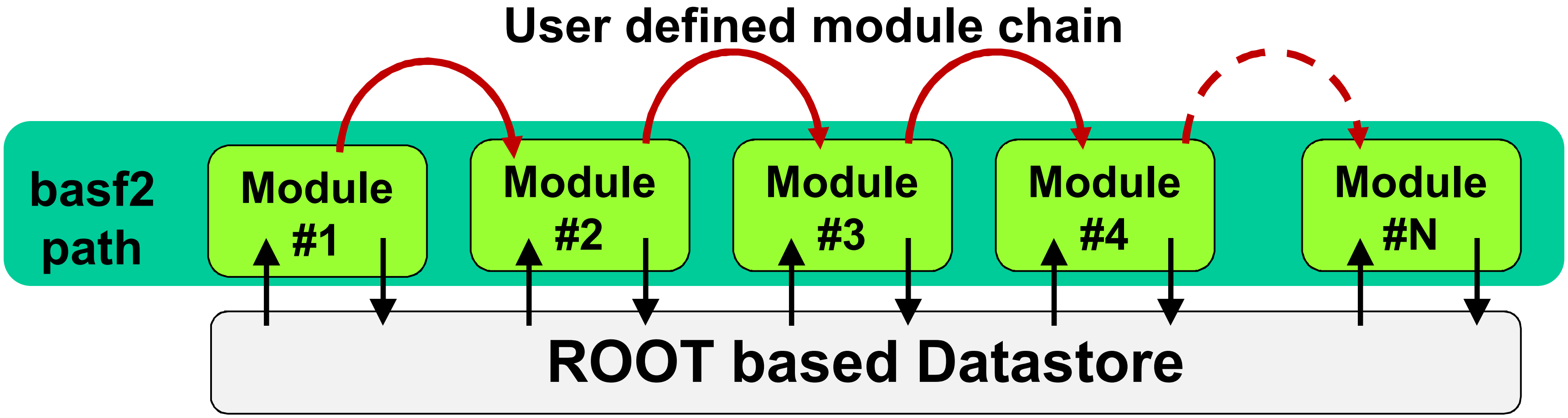
- Old system: C++/C/Fortran. Custom storage format to store data.
- New system: C++ based. ROOT I/O
- Imported useful concepts from other experiments:
  ILC, LHCb, CDF, and Alice



- Third-party software libraries:
  Geant4, ROOT, boost, CLHEP, and many others.

- Software developers are from all around the world:
  Weekly developers' meetings, two workshops per year.

The Basic Structure of BASF2



- **Module**: The basic processing unit
  - Examples: As simple as reading data from a file
    to complex ones like simulation or tracking
  - All the works are done in modules.
  - Selection and arrangement of the modules are done by a user.
- A typical event processing = a linear chain of modules on a **path**
- **Datastore**: A common storage for data

Example of Python Steering File

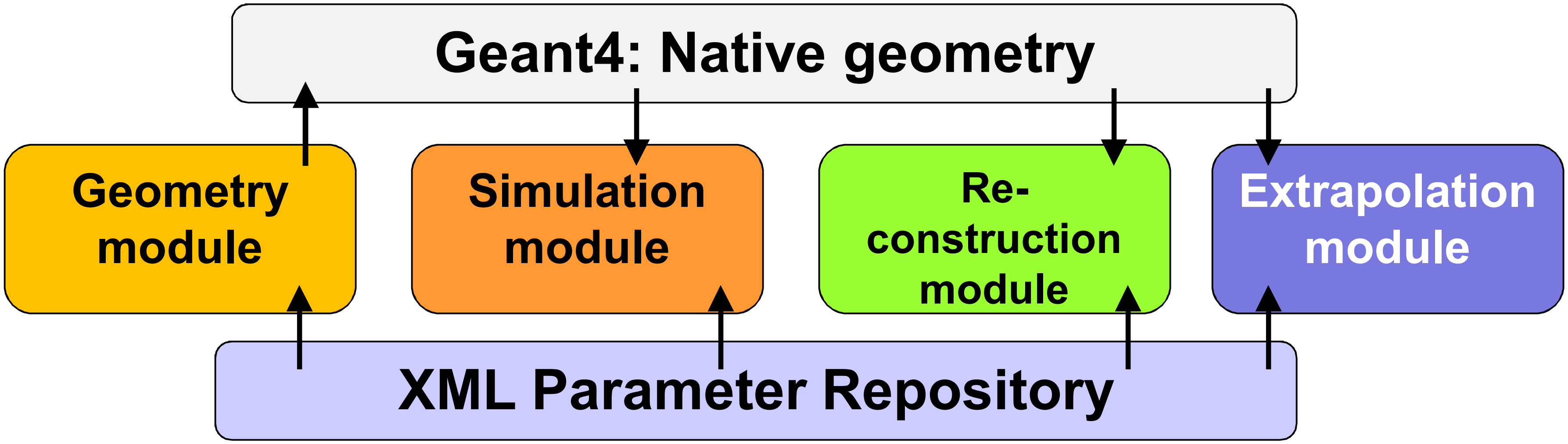
Main Steering File

```
from basf2 import *
main = create_path()
# event meta data
main.add_module(event_meta_data)
# generator (evtgen,  particle gun, etc.)
main.add_module(evtgen)
# simulation
main.add_module(simulation)
# reconstruction
main.add_module(reconstruction)
# output
main.add_module(output)
process(main)
```

Simulation Steering File

```
from basf2 import *
# geometry parameter database
path.add_module(gearbox)
# detector geometry
path.add_module(geometry)
# background mixing
path.add_module(beam_background)
# detector simulation
path.add_module(geant4_interface)
# PXD simulation (digitization, clustering )
path.add_module(PXD)
# SVD simulation (digitization, clustering)
path.add_module(SVD)
# Other sub-detectors here.
```

Geometry Handling System

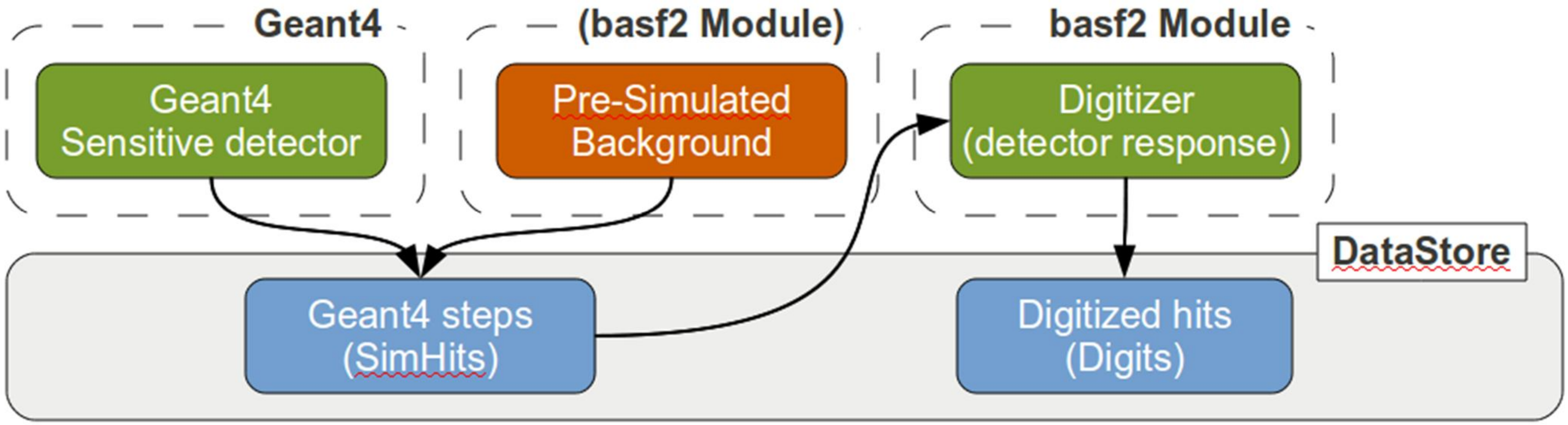


- All the geometry parameter values are stored in the central repository. Plan to move from the XML format to a database.
- The actual geometry for simulation is created from the repository parameters via C++ algorithms.
- For event display, it is converted to ROOT **TGeo** via VGM library.

Many Event Generators are Supported

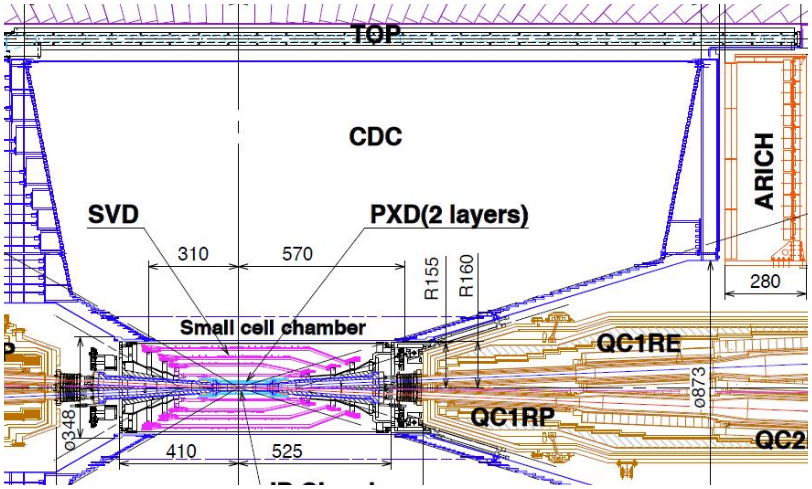
- EvtGen 1.4.0 + TAUOLA, PHOTOS and Pythia 8 interfaces.
- Precision QED libraries:
  - PHOKHARA 9.1, KKMC, BHWIDE, BabayagaNLO, etc.
- Two photon physics: AAFH, KORALW, etc.
- Dark photon and exotic events: MADGRAPH
- Cosmic ray: CRY
- Debugging and testing: ParticleGun

Background Mixing



- Pre-simulated Geant4 background hits are added as SimHits (Geant4 steps) to the already existing SimHits from the signal event.
- The background event could be from Touscheck, radiative Bhabha, beam-gas, beam-wall, etc.
- Then both contributions are merged and **digitized** at the same time.

Offline Tracking Reconstruction Algorithm

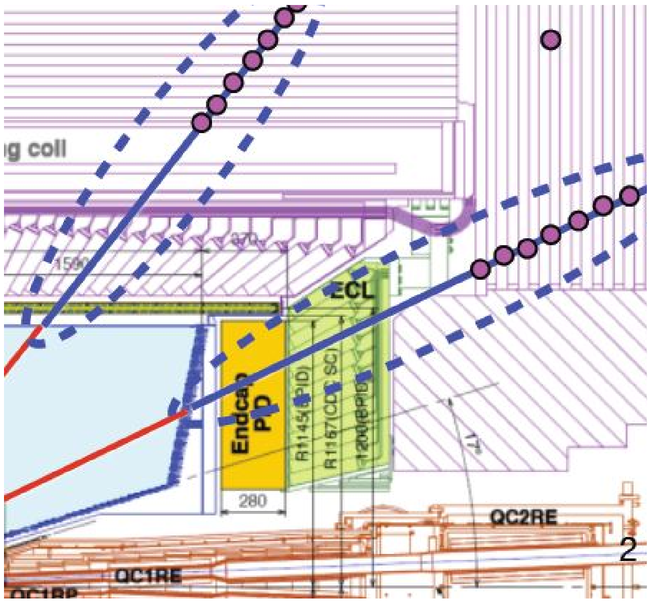


1. Standalone finder for PXD + SVD:
   - Cellular automaton
   - Hopfield-Network finder
2. Standalone finder for CDC:
   - Primary tracks: Legendre finder
   - Secondary tracks, decays in flight, cosmic tracks: Weighted cellular automaton,
3. Merging of 1 + 2
4. (Planned: Cross detector searches and extrapolations for additional hits)
5. Final Fit: GENFIT Kalman Filter from seed delivered by the finders.

PID Tools

Tools in development:

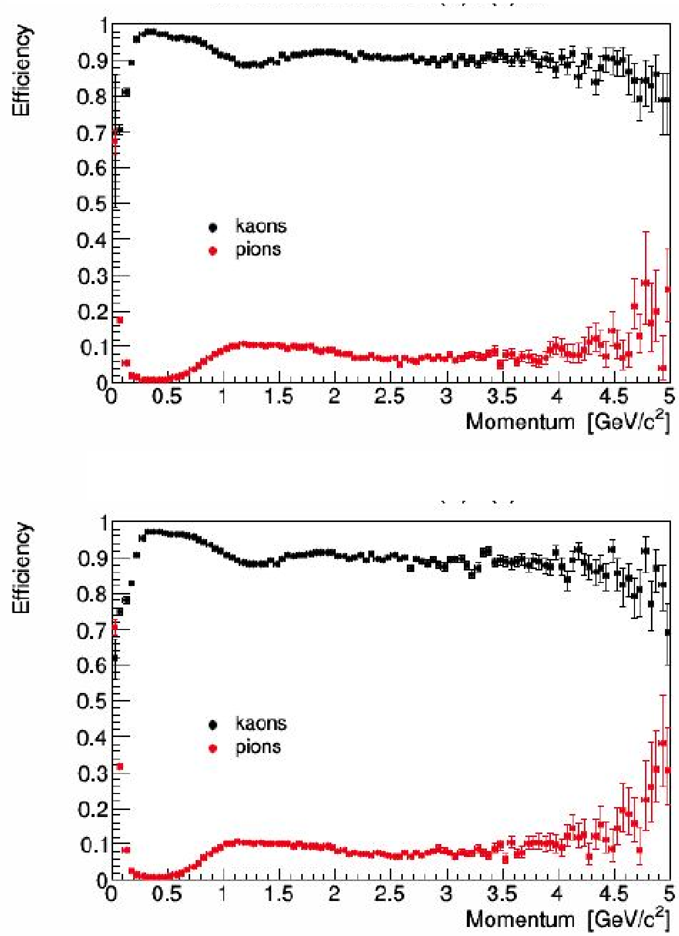
- Neutrals (π^0, photon, K_L, K_S)
- Charged (e, μ, $\pi^{\pm}$, $K^{\pm}$, p)$_{,}$ deuteron



Central track + muon hits with Kalman filter

K/π separation

TOP+ARICH+dE/dx



no BG

BG ×1

Analysis Tools for Physics

Analysis Steering File

```
# other modules at the beginning.
# reconstruction
main.add_module(reconstruction)
# create final state particle lists
kaons = (‘K-’, ‘’)
fillParticleLists([kaons, pions, elecs, photons], main)
# reconstruct D0 -> K- pi+ decay (and c.c.)
reconstructDecay('pi0 -> gamma gamma', '0.05 < M < 1.7', main)
# reconstruct Btag -> D0 pi- (and c.c.)
reconstructDecay('B-:tag -> D0 pi-', '5.000 < M < 6.000', main)
# reconstruct Bsig -> pi0 e+ [nu_e] (and c.c.)
reconstructDecay('B+:sig -> pi0 e+', '0.000 < M < 6.000', main)
# reconstruct Y(4S) -> Btag Bsig
reconstructDecay('Upsilon(4S) -> B-:tag B+:sig', '0.000 < M < 11.000',
main)
```

- Analysis is done in modules.
- Data Model:
  - Class Particle:
  - StoreArray <Particle>
  - Class ParticleList
- Tag-side B reconstruction is possible

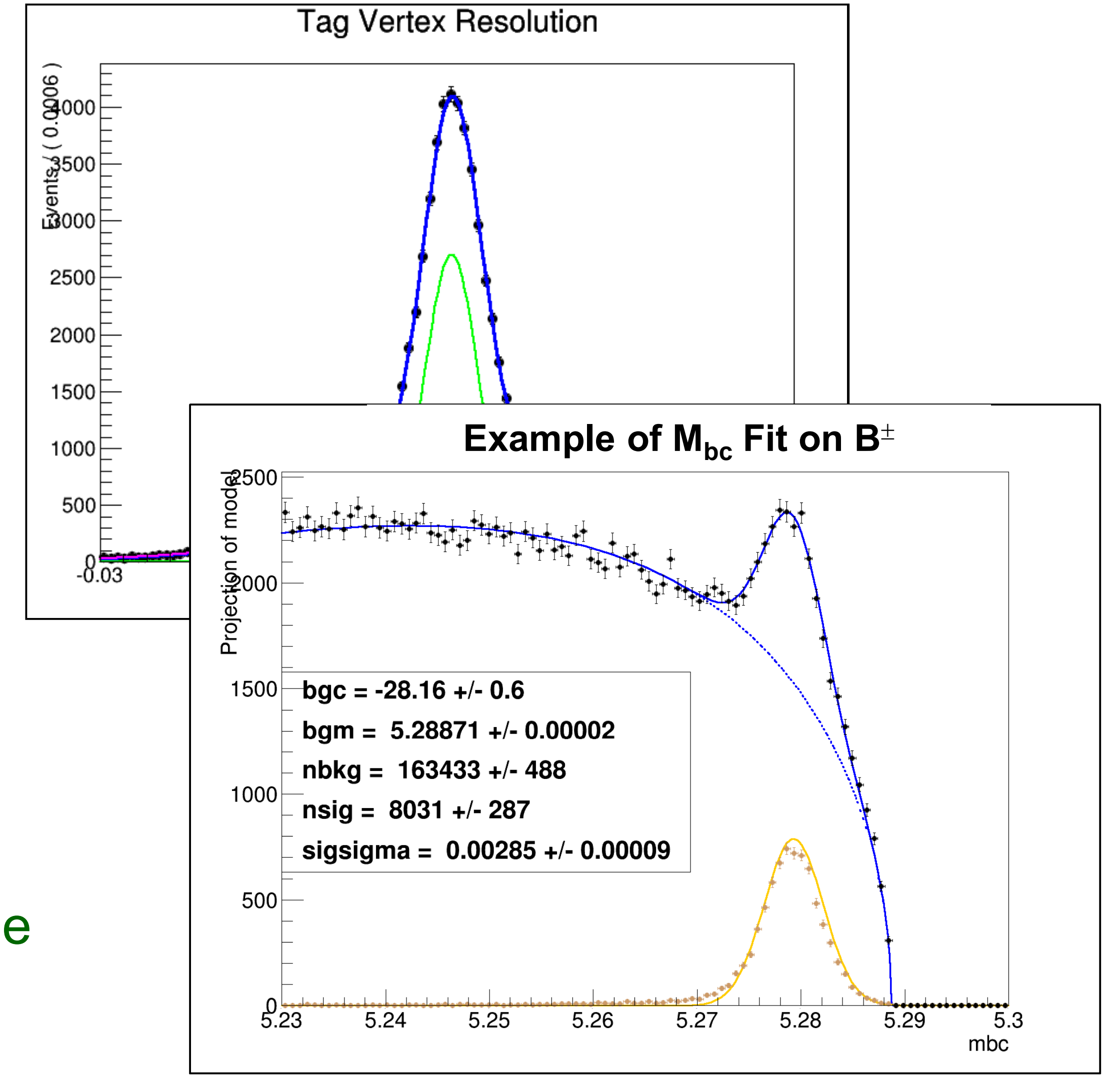
Analysis Tools for Physics Continues…

Basic modules:

- ParticleLoader
- ParticleSelector
- ParticleCombiner
- VertexFitter

Other modules

- Best candidate selection
- MC truth matching
- Continuum suppression
- TMVA (neural network)
- (B) Flavor tagging.
- Rest-of-event to a particle


Tag Vertex Resolution

Example of M_{bc} Fit on $B^{\pm}$

And additional data objects.

Summary

- Belle II Experiment, as of July 2015,
  Over 600 scientists. 98 institutions. 23 countries/regions.

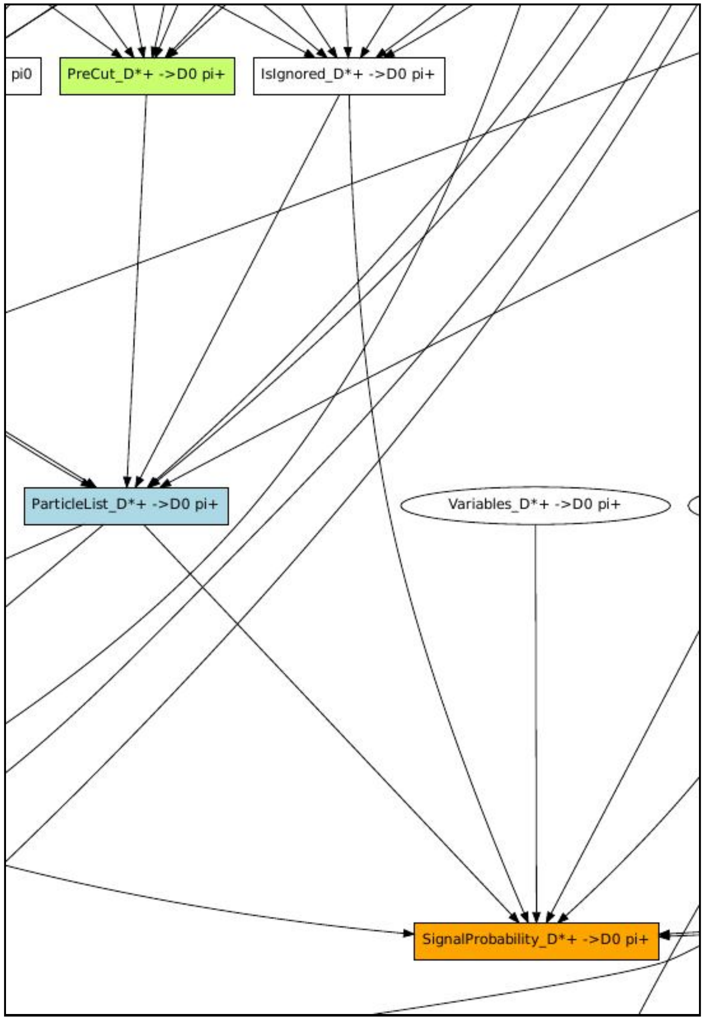
- Active development in software:
  Many ideas are tested and merged into stable library packages (but could not be covered in this talk).
  Physics analysis is now possible.

- Regular (twice per year) runs of MC mass production & data transfer challenge test (Dr. Vikas Bansal's talk next).

- The basf2 framework = ~650k lines of C++ code (excl. comments)
  + Python, external SW libraries, etc.

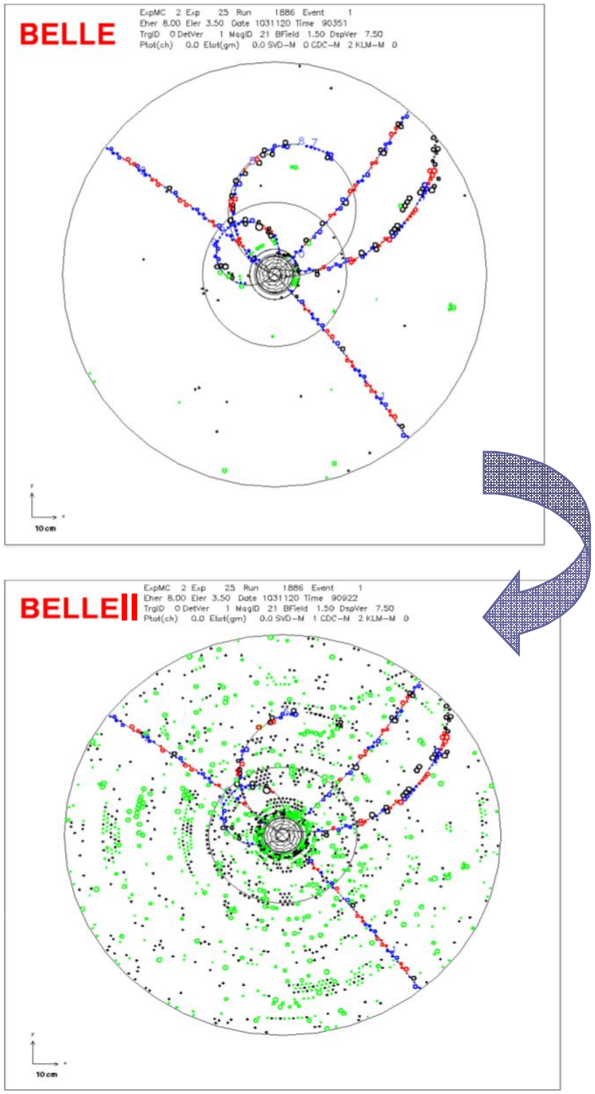
- Many thanks for the colleagues who provided valuable ideas, data and plots for this talk.

Extra Slides

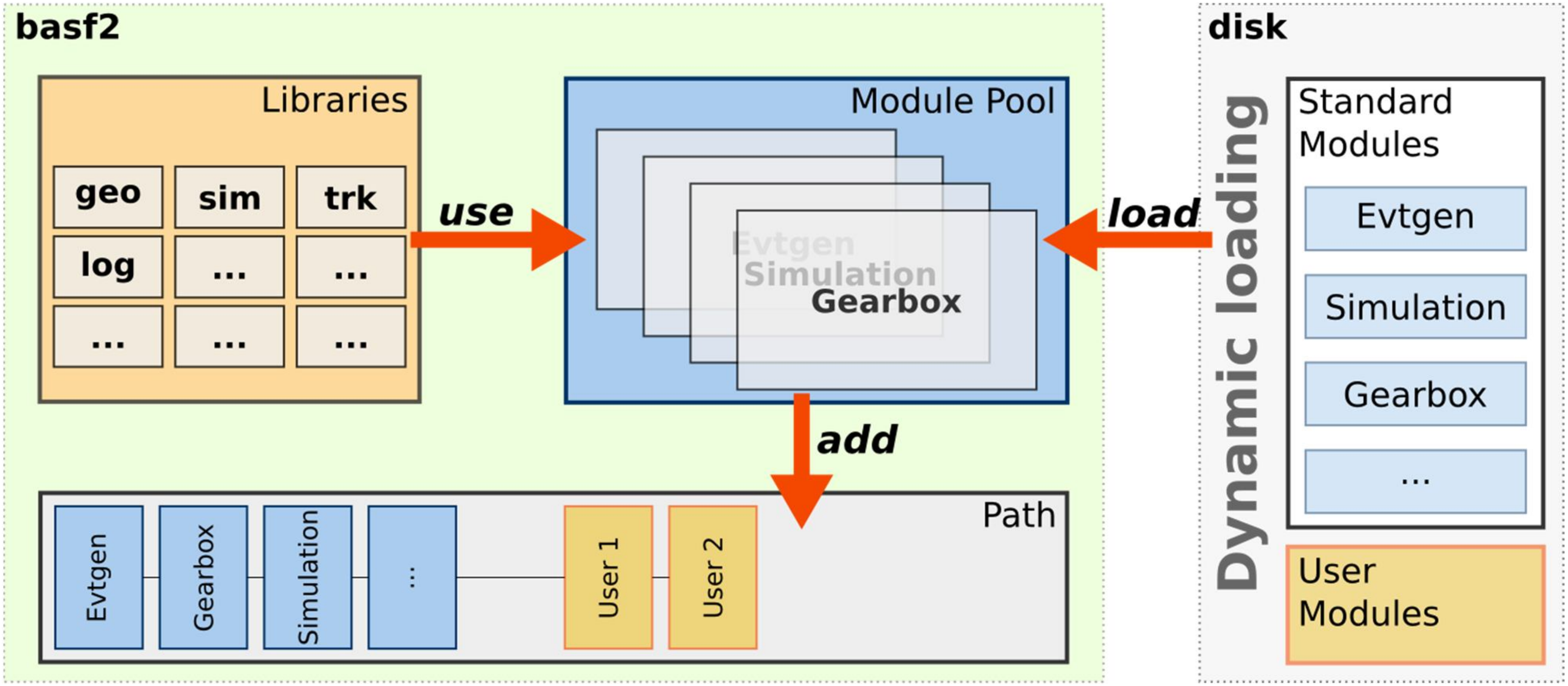


Challenges to Belle II

- Higher background (× 10~20 of KEKB).
  - Touscheck scattering
  - Radiative Bhabha
  - 2-photon
  - More radiation damage, fake hits, pile-up
- Higher event rate (×10)
  - L1 trigger rate: ~20 kHz
- Improvements planned:
  - Better hermeticity.
  - Better IP and secondary vertex resolution.
  - Better PID.

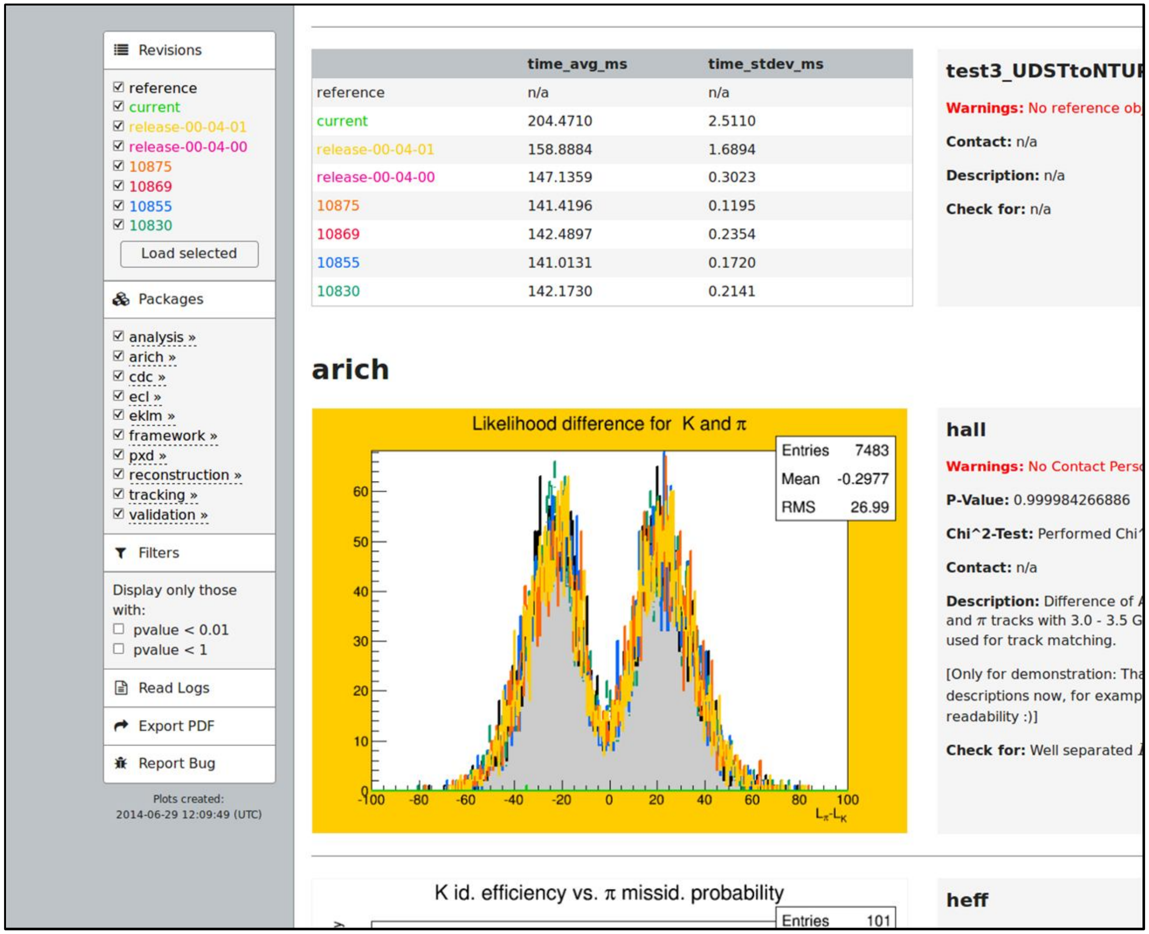


Libraries vs Modules



- **Libraries:** Separated from modules to increase reusability.
  - Methods and algorithms are encapsulated in libraries.
  - A library (i.e, algorithm) can be used/shared by several modules.

Validation of basf2 : Example



	time_avg_ms	time_stdev_ms
reference	n/a	n/a
current	204.4710	2.5110
release-00-04-01	158.8884	1.6894
release-00-04-00	147.1359	0.3023
10875	141.4196	0.1195
10869	142.4897	0.2354
10855	141.0131	0.1720
10830	142.1730	0.2141

test3_UDSTtoNTUF

Warnings: No reference ob

Contact: n/a

Description: n/a

Check for: n/a

arich

hall

Warnings: No Contact Perso

P-Value: 0.999984266886

Chi^2-Test: Performed Chi^

Contact: n/a

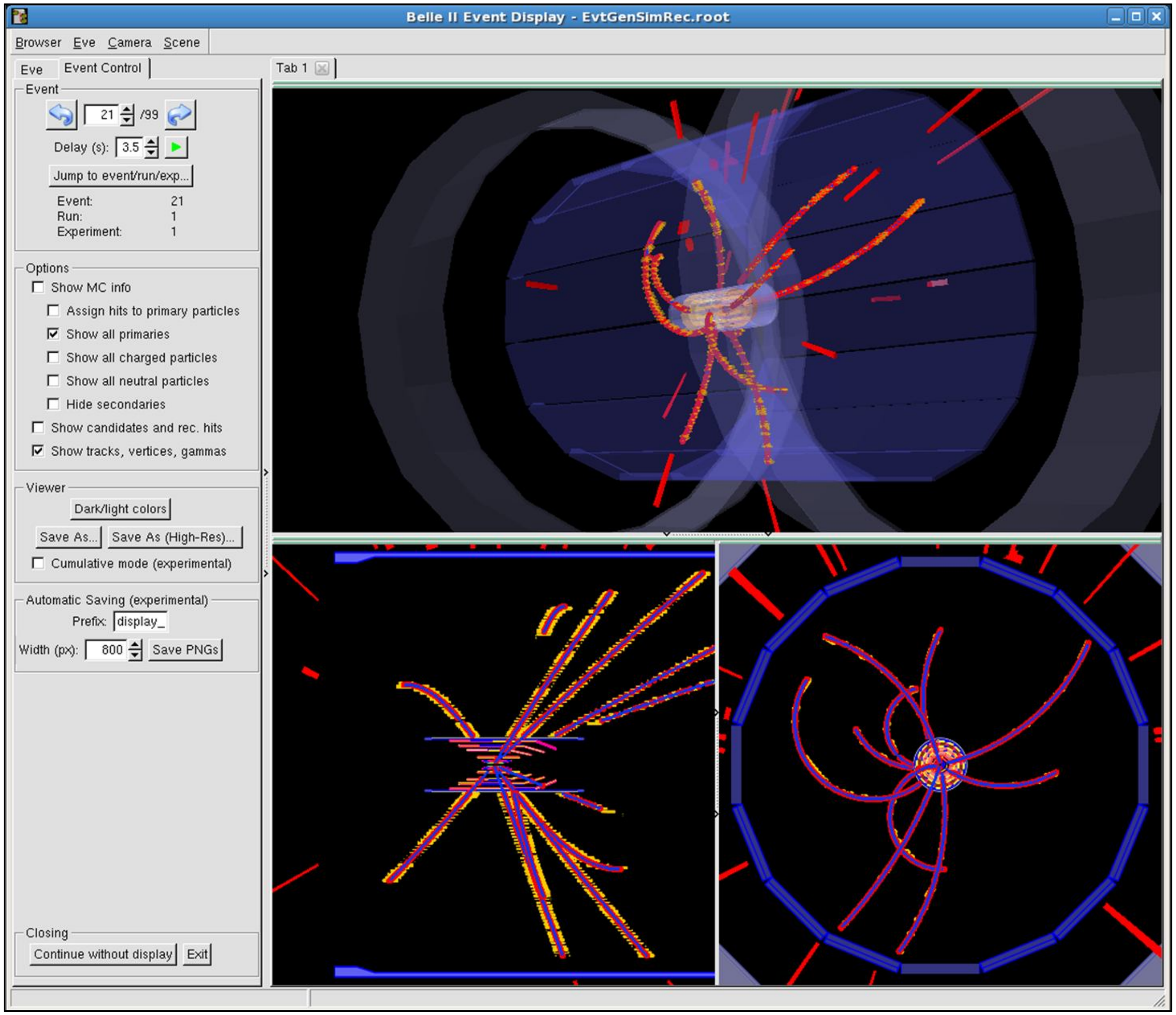
Description: Difference of A and π tracks with 3.0 - 3.5 G used for track matching.

[Only for demonstration: Tha descriptions now, for examp readability :)]

Check for: Well separated

heff

Event Display



Event

21 /99

Delay (s): 3.5

Event: 21
Run: 1
Experiment: 1

Options

- ☐ Show MC info
  - ☐ Assign hits to primary particles
  - ☑ Show all primaries
  - ☐ Show all charged particles
  - ☐ Show all neutral particles
  - ☐ Hide secondaries
- ☐ Show candidates and rec. hits
- ☑ Show tracks, vertices, gammas

Viewer

☐ Cumulative mode (experimental)

Automatic Saving (experimental)

Prefix: display_

Width (px): 800

Closing

Basf2 + ROOT with OpenGL support