The Belle II Software Framework and ROOT

ROOT Users Workshop 2013
Christian Pulvermacher | March 13th 2013
The Belle II Experiment

- Experiment at the SuperKEKB B factory being built in Tsukuba, Japan
- Successor of Belle (1999–2010), with $40 \times$ higher luminosity and upgraded detector
- Planned start in 2016
Software Framework

**Requirements**
- Object-based I/O, easy integration with analyses
- Same framework for offline and online reconstruction (DAQ, HLT)
- Event-by-event parallel processing
- Compatibility with Belle software

Some options evaluated:
- Belle software (basf)
- ILC software framework
- ALICE software framework
- GAUDI
Software Framework

Requirements

- Object-based I/O, easy integration with analyses
- Same framework for offline and online reconstruction (DAQ, HLT)
- Event-by-event parallel processing
- Compatibility with Belle software

Some options evaluated:

- Belle software (basf)
- ILC software framework
- ALICE software framework
- GAUDI

Different ideas combined in new software framework specific to Belle II (basf2).
Overview of basf2

- Written in object-oriented C++
- Modular simulation / reconstruction / analysis toolchain
- ROOT I/O for (de)serialisation of objects

Used libraries
- ROOT
- boost
- EvtGen
- Geant4
- GenFit
- VGM
+ some others.
Python Steering

Steering files are just Python scripts:

```python
from basf2 import *
main = create_path()

# more modules here ...

particlegun = register_module('ParticleGun')
particlegun.param('pdgCodes', [-13, 13])
particlegun.param('nTracks', 4)
main.add_module(particlegun)

process(main)
```

- Implemented using boost::python
- Scripts can be quite simple – but also powerful, with different paths for different events, user input, plotting, etc.
Most modules will be implemented in C++, but it’s also possible with Python:

```python
class MinModule(Module):
    def __init__(self):
        super(MinModule, self).__init__()

    def event(self):
        simhits = Belle2.PyStoreArray('PXDSimHits')
        for hit in simhits:
            mcpart = hit.getRelatedFrom('MCParticles')
            mcpart.Dump()
```

- boost::python for calling Python functions from C++
- PyROOT for access to data shared between modules.
- Very useful for debugging, tests, creating plots
Parallel Processing

- Event based
- No threads (would require additional effort, libraries mostly not thread-safe)
- Processes are fork()ed, data exchanged through shared memory (using TMessage)
Event Display Module

- Developed using Eve
- Geant4 geometry converted to TGeo
- Planned to be used for online monitoring
Things that work well

We’ve been quite happy with

- ROOT I/O and introspection
- PyROOT
- EVE
- Feature range of histograms, plotting, etc.
Things that work well

We’ve been quite happy with

- ROOT I/O and introspection
- PyROOT
- EVE
- Feature range of histograms, plotting, etc.

ROOT developers are also fairly responsive to bug reports.
What could be improved

- Magic deletion (e.g. with TFile)
- Double ownership (e.g. TEveGeo objects)
- Unclear ownership
- Class documentation often not enough; user manual outdated
- Functions/arguments rarely \texttt{const} when they could be

We’re looking forward to ROOT6 and support of STL and C++11.
What could be improved

- Magic deletion (e.g. with TFile)
- Double ownership (e.g. TEveGeo objects)
- Unclear ownership
- Class documentation often not enough; user manual outdated
- Functions/arguments rarely `const` when they could be

We’re looking forward to ROOT6 and support of STL and C++11.

**Basf2 framework developers**: Martin Heck (KIT), Ryosuke Itoh (KEK), Thomas Kuhr (KIT), Andreas Moll (MPI), Christian Pulvermacher (KIT), Martin Ritter (MPI)