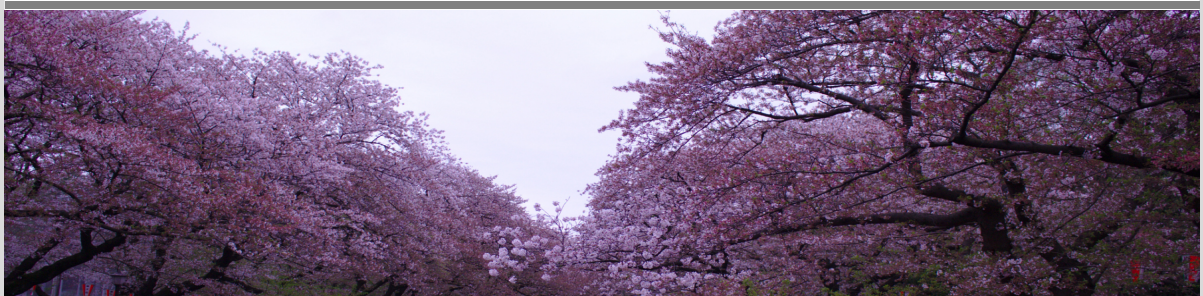
An Interactive and Comprehensive Working Environment for High-Energy Physics Software with Python and Jupyter Notebooks

CHEP 2016 – San Francisco, United States of America

Nils Braun[1], Thomas Hauth[1], Christian Pulvermacher[2], Martin Ritter[3]
1 **Institute for Experimental Nuclear Physics (IEKP), KIT, Germany**
2 **KEK, Tsukuba, Japan**
3 **Ludwig-Maximilians-University, Munich, Germany**



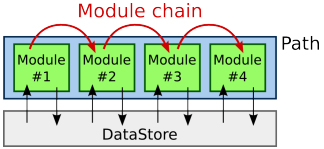
SuperKEKB and the Belle II Experiment

- KEKB was an electron–positron collider in Tsukuba/Japan which studied the decay of B mesons at the Y(4S) resonance
- Discovery of CP violation in the B system by the Belle experiment at KEKB and by the BaBar experiment at SLAC
  - Nobel Prize in Physics to Kobayashi and Maskawa in 2008
- The SuperKEKB collider and the Belle II detector will build on the previous success:
  - Study B meson system in far greater precision
  - Probe for new physics in a wide range of interesting topologies

	KEKB	SuperKEKB	Factor
Instantaneous Luminosity	$2 \times 10^{34}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$	$8 \times 10^{35}\,\mathrm{cm}^{-2}\mathrm{s}^{-1}$	40
Integrated Luminosity	$1\,\mathrm{ab}^{-1}$	$50\,\mathrm{ab}^{-1}$ (projected)	50
Runtime	1998 to 2010	start in 2017	-
Detector	Belle	Belle II	-
Raw Data	1 PB	100 PB (projected)	100

The `basf2` Framework

- Event processing framework for Belle II written in C++11
- Modules use a common data store to read event data and write back results
- All processing steps for recorded and simulated events are implemented in basf2: Event generation, simulation, digitization, HLT, reconstruction and analysis
- Important libraries are bundled into *externals*: ROOT, gcc, Geant4, ...



Python is a first-class citizen in our framework:

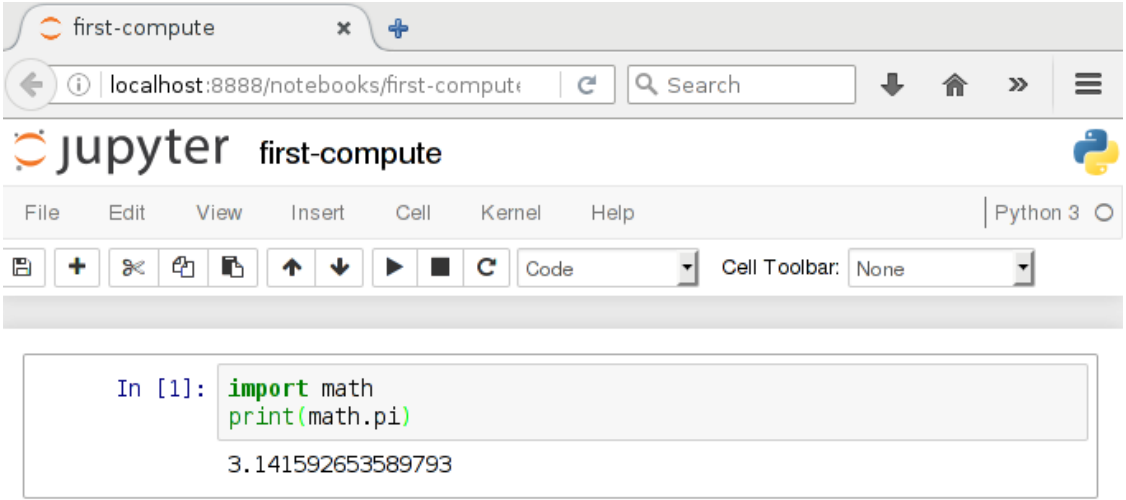
- Steering files connect modules to paths and are written in Python 3
- Modules can be written in C++ and Python
- Framework functionality is exported to Python via `boost::python`
- User classes and all objects within the **DataStore** are available in Python via **PyROOT**

Jupyter Notebooks

Jupyter inherits all benefits available to Python in our framework, plus:

- Notebooks
  - Save expressions and corresponding results in one place
  - Include comments, documentation, pictures, drawings, LaTeX, videos
  - Send notebooks, including all results, to someone else (use-cases: software examples, bug reports)
  - Perform analyses interactively
- Clickable widgets in HTML and JavaScript
- Sections of a notebook can be executed individually
- Tab-completion and syntax highlighting
- Server–client structure
  - Access the Jupyter service from your home computer, smartphone, tablet, etc. but run the calculations on a high-performance machine
  - No need to rely on X forwarding or other technologies
- Many data science tools with Jupyter integration: ROOT, matplotlib, pandas
- Not only for Python (Haskell, Julia, C++, ROOT, Terminal)

Jupyter Notebooks - Example



Under the Hood: Integrating basf2

We developed the Python library `hep_ipython_tools` [1] which simplifies the integration of HEP Frameworks with jupyter notebooks.

Core component for seamless Jupyter integration of `basf2`:
Process handler for background framework execution

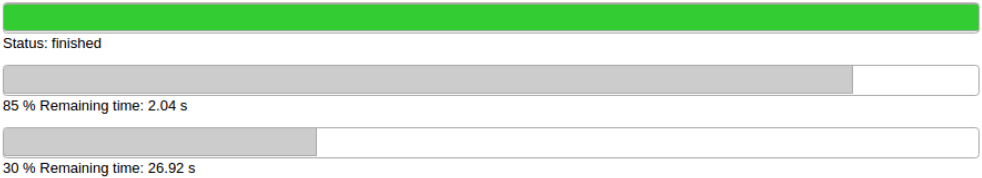
- Creates a separate worker process for `basf2`
- Transfers path configuration and starts processing
- Monitors running framework process
- Installs a message queue between jupyter and basf2 processes to transfer status information (current event number, performance statistics etc.)
- Can support multiple basf2 Instances to concurrently scan a parameter space
- Implementation is generic and can be easily adapted to support other frameworks

Better User Experience: Widgets

- Jupyter Widgets are graphical exentsions to notebooks to better view use-case specific contents
- Written in Python and JavaScript, running interactively in the user's browser
- Allows to use rich library ecosystem of Python and web-development world (jQuery, HTML5, CSS etc.)

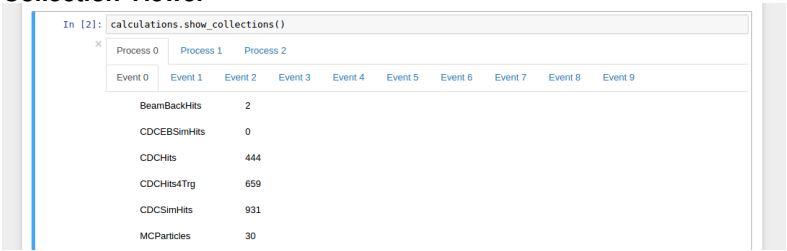
We developed a set of Jupyter widgets to improve the user experience of basf2 in Notebooks

Progress Bar

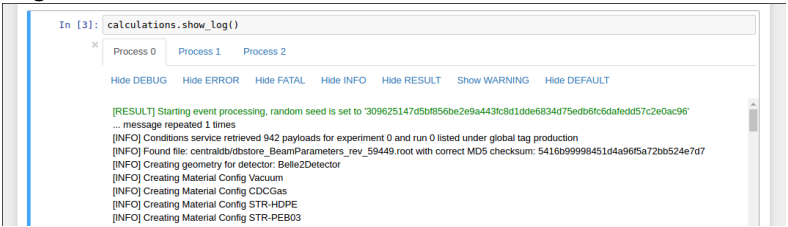


Better User Experience: Widgets

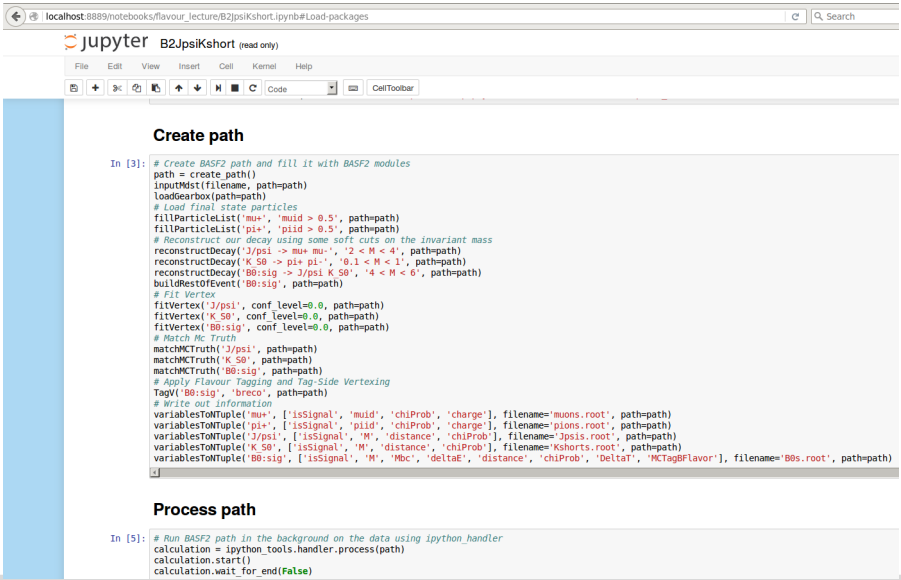
Collection Viewer



Log Parser



Full Analysis Example



Full Analysis Example

Read input files

In [6]:
```
Jpsis = root_pandas.read_root('Jpsis.root')
Kshorts = root_pandas.read_root('Kshorts.root')
B0s = root_pandas.read_root('B0s.root')
B0s.describe()
```

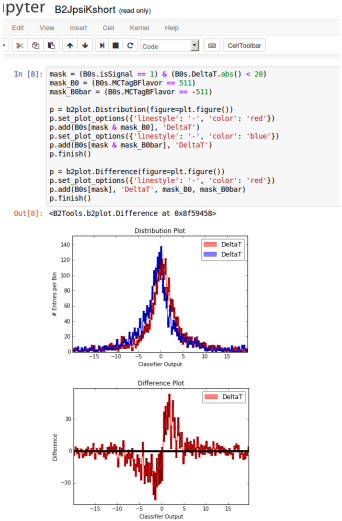
Out[6]:

	isSignal	M	Mbc	deltaE	distance	chiProb	DeltaT	MCTagBFlavor
count	37777.000000	37777.000000	37777.000000	37777.000000	37777.000000	37777.000000	37777.000000	37777.000000
mean	0.117029	4.831285	5.028489	0.251113	1.872794	0.189709	6.236745	-4.287980
std	0.321459	18.471546	0.582552	38.040882	8.319394	0.302396	752.426514	510.447388
min	0.000000	1.213630	0.000000	-3.947679	0.000476	0.000000	-31209.353516	-511.000000
25%	0.000000	4.193513	5.008841	-0.796031	0.026089	0.000000	-5.144518	-511.000000
50%	0.000000	4.499523	5.141731	-0.536930	0.088729	0.000221	-0.027970	-511.000000
75%	0.000000	5.079098	5.241263	-0.082886	0.590760	0.305392	3.952079	511.000000
max	1.000000	3556.052979	5.286911	5543.626953	270.880585	1.000000	31147.080078	511.000000

Do some plots

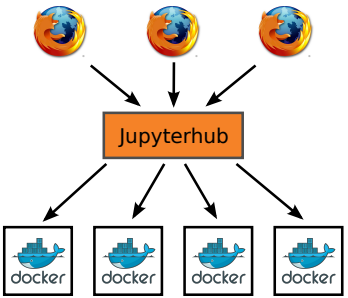
In [7]:
```
p = b2plot.Distribution(figure=plt.figure())
p.set_plot_options({'linestyle': '-', 'color': 'red'})
p.add(Jpsis[(Jpsis.M > 2) & (Jpsis.M < 4) & (Jpsis.isSignal == 1)], 'M')
p.set_plot_options({'linestyle': '-', 'color': 'blue'})
p.add(Jpsis[(Jpsis.M > 2) & (Jpsis.M < 4) & (Jpsis.isSignal == 0)], 'M')
p.finish()
```

Full Analysis Example



The shown notebook was already successfully tested with students in a tutorial at KIT.

The Big Picture



Your browser is the UI

Central webservice

Distributed Computing

- Analysis in the cloud (analysis as a service)
- The notebooks can be used for outreach (e.g. tutorials in universities and schools)
- Jupyterhub provides a jupyter notebook server with authentication, user management, distributed computation/cluster support.
- Prototype and evaluation setup is running successfully on a machine at KIT
- In the future, we hope that data centers offer to host this kind of services

Conclusion and Outlook

- Perform Python calculations with Jupyter notebooks to have all benefits of Python together with the interactivity.
- The lightweight software layer provided by `hep_ipython_tools` allows a **seamless integration of HEP frameworks** (here basf2) with interactive jupyter notebooks

  Notebooks can be used for:
  - Interactive development of framework module algorithms
  - Working on analyses with fast feedback via inlined plots
  - Self-describing Notebooks for tutorials and outreach
- Using jupyter(hub) with basf2 is a full environment for physics analysis!
- In the future: possibilities for interactive physics analysis via the web browser, centrally hosted at data centers

References I


“HEP IPython Tools.”
https://github.com/hep-ipython-tools/hep-ipython-tools (28.9.2016).