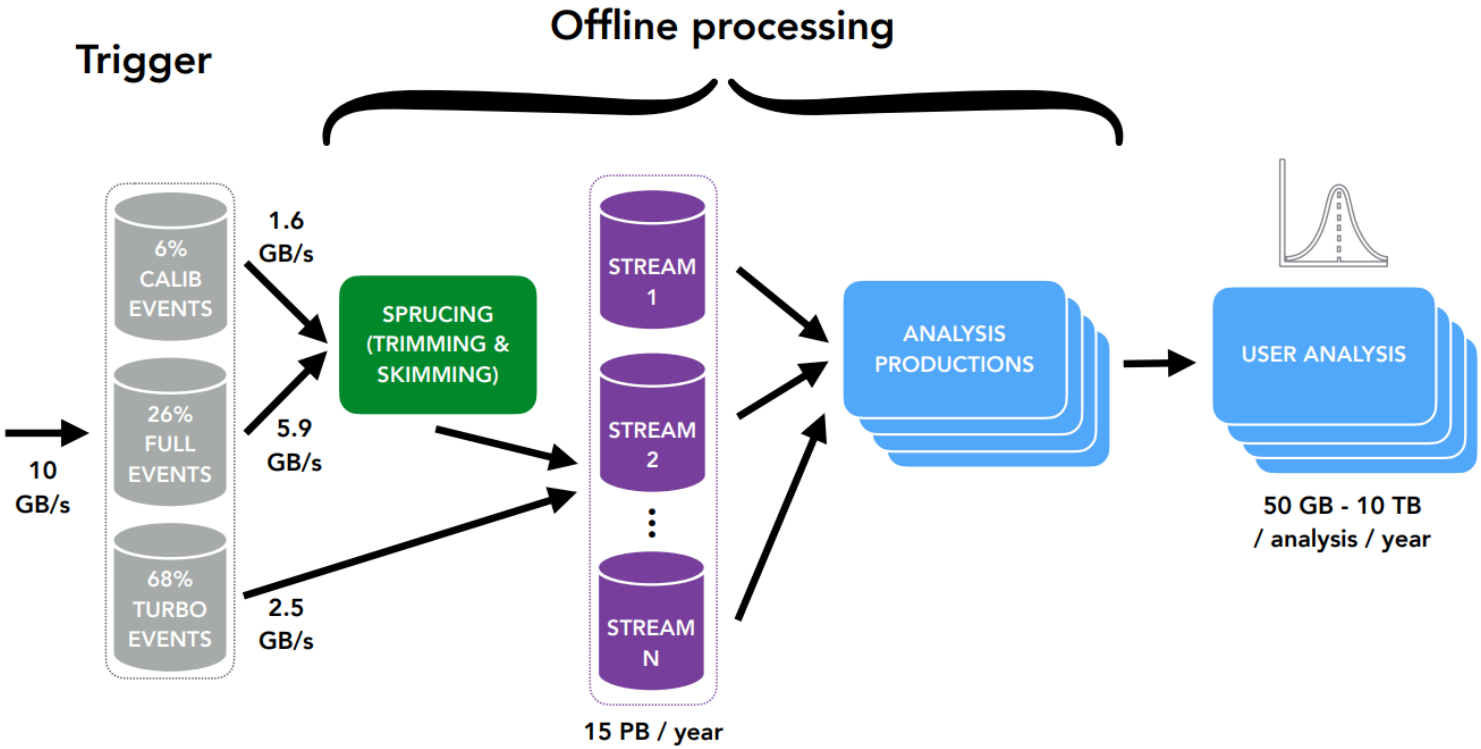
Python Usage Within the LHCb Experiment - PyHEP 2022 Workshop

Nate Grieser

University of Cincinnati, on behalf of the LHCb Collaboration

14-09-2022

Data Workflow in LHCb



- Large infrastructure for data processing, primarily C++, Python steering
- Many tools available with the ability to complement the structure!

The tools in today's discussion focus on the last two stages!

▸ LHCB-FIGURE-2020-016

GOAL: Share the Python tools and experiences of analysers from the LHCb collaboration.

Analysis in LHCb

Data Format: LHCb analyses primarily use flat nTuples outside of the physics application stack

- Filtering of samples to create nTuples done with individual grid jobs → Centralised with Run 3

SnakeMake: Utilized to create reproducible and scalable analyses.

Gitlab CI: Interest to run analyses using this tool → concerns with data access and caching

Software Environments: LHCb employs the use of environments to run analyses

- Past:
  - Interactive clusters: abusing physics application environments and using central CERN building-blocks (LCG)
  - Locally: Ad-hoc builds, homebrew, system package managers
- Run 3 and beyond:
  - Interactive clusters: Versioned conda environments installed on CVMFS and centrally managed
  - Locally: Conda/mamba with conda-forge

Analysis Productions in LHCb

Centralized sample filtering to create manageable nTuples for analysers to use; tools used in checks and validation

uproot

- I/O for ROOT files: better performance than ROOT's own file I/O methods
- Allows to read the data straight into arrays/dataframes

awkward, numpy, pandas

- Physics tool manipulation handling before shipping to the nTuple outputs
- Flexible use with uproot to change between formats during development

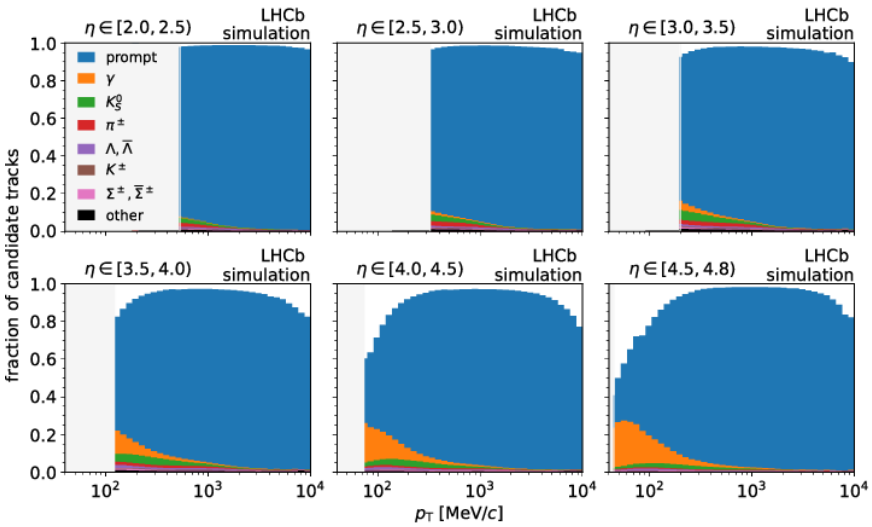
Feedback:

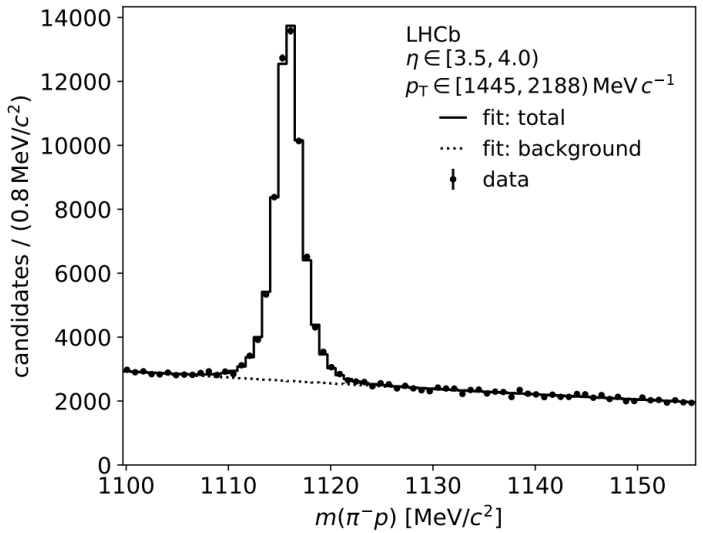
- Significant improvement in the reading of files with pythonic tools $\rightarrow$ huge performance boost when handling many files
- Pythonic dataframe structures integrate more nicely into other libraries rather than using ROOT's custom versions
- Some missing functionalities (e.g. JSON serialisation for hist) but can be manually implemented reasonably
- Lack of documentation in some libraries makes difficult $\rightarrow$ need expert understanding to trawl source code

LHCb Publication Using Solely Scikit-HEP Tools

Post data-processing all performed with Python HEP tools!

- Uproot: Interfacing with input ROOT files
- boost-histogram: Replace classic ROOT classes; Bonus: Multi-dimensional histograms!
- iminuit: User-friendly interface to minuit2 to process minimization
- PDF build in Python using SciPy library components





▸ LHCb-PAPER-2021-010

LHCb Measurement - Feedback From Users

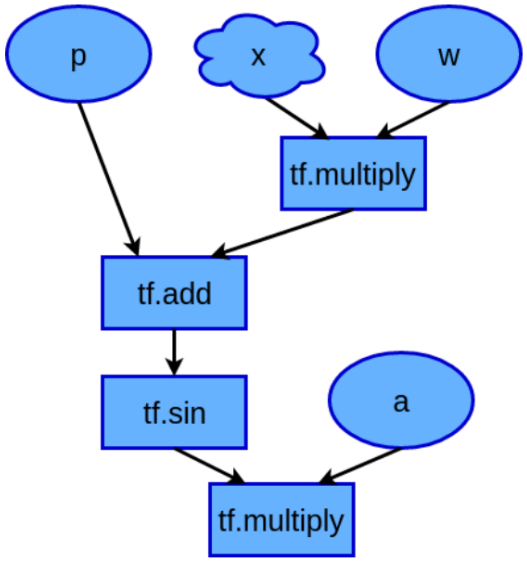
- Standard scientific Python stack can handle a good portion of the analysis flow → Still need a splash from other HEP tools
- Python tools often are more lightweight (uproot) and flexible (boost histogram) than their ROOT counterparts.
- Python packages often have a simpler implementation and lower learning curve → great for getting students involved
  - About 50% of LHCb analysers using Python-based analysis code!
- Many users often try alternatives, but fallback to RooFit when other fitting tools aren't mature enough for non-trivial fits.
- Bootstrapping availability was missing up until recently → resampling package added to Scikit-HEP.
- Architecture can still benefit from a general purpose error propagation library → Jacobi library to possibly address this gap.

Amplitude Analysis Tools in LHCb

Amplitude Analyses are the most computationally heavy analyses in the LHCb physics program. Many tools available to fill this niche, focus on one today, but many more on the market:

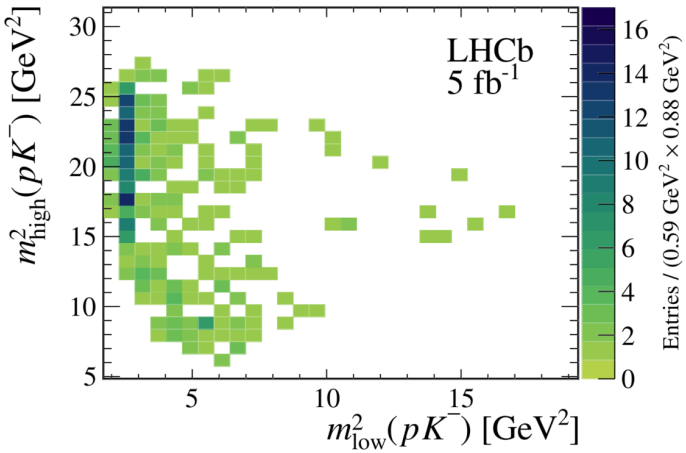
Tensor Flow:

- Open-source computing software provided by Google
- Serves as a low-level library to provide building blocks of complex fitting
- Number of institutes making use of GPU availabilites for fitting, DNNs, and amplitude analyses
- Additional potential to exploit the large LHCb HLT1 GPU farm outside of data-taking



AmpliTF:

- Library of simple HEP-applicable functions using the TF building blocks.
- Simplified class structure allows for accessibility to developers and application to other libraries.



▸ LHCb-PAPER-2020-017

AmAna Tools - Feedback From Users

- Primary goal: When a framework isn't readily available, need efficient computation with a tradeoff between time to implement the code, and the processing time it saves when running.
- Frameworks like zfit do this: iminuit minimization backend, TF backbone hidden from users, flexible
- TF is heavy, and not always hardware-accessible $\rightarrow$ continuing to look beyond TF (JAX, pyTorch, etc.)
- When developing code and frameworks using TF, need to keep it as a building block, and not a cornerstone $\rightarrow$ Allows to easily migrate to another computation engine in the future.

zfit - General Purpose Fitter

General purpose fitter to harmonize workflows and include advanced functionality beyond what is provided by standard Python tools.

- TensorFlow: Used as a backend for parallelization and GPU support
- Additionally makes use of uproot, boost-histogram, and iminuit
- hepstats: Used in conjunction with zfit for further statistical inferences.

- Roughly 100 users across LHCb and Belle 2 experiments

Two primary analysis use cases for the package:

1. Simple 1D mass fits → pythonic package allows for easy implementation
2. Extremely complicated fits → extremely malleable, making many fits possible!

▸ zfit Github ▸ LHCb-PAPER-2016-019

$$\mathcal{M}^{K^*}_{\Delta\lambda_\mu} \equiv \sum_j R_j(m_{\phi K}) \sum_{\lambda_{J/\psi}=-1,0,1} \sum_{\lambda_\phi=-1,0,1} A^{B\to J/\psi\, K^*\, j}_{\lambda_{J/\psi}}\, A^{K^*\to\phi K\, j}_{\lambda_\phi} \times$$
$$d^{J_{K^*}\, j}_{\lambda_{\mathrm{J}/\psi},\lambda_\phi}(\theta_{K^*})\, d^1_{\lambda_\phi,0}(\theta_\phi)\, e^{i\lambda_\phi \Delta\phi_{K^*,\phi}}\, d^1_{\lambda_{\mathrm{J}/\psi},\Delta\lambda_\mu}(\theta_{J/\psi})\, e^{i\lambda_{J/\psi}\Delta\phi_{K^*,J/\psi}}$$
$$|\mathcal{M}^{K^*}|^2 = \sum_{\Delta\lambda_\mu=\pm1} \left|\mathcal{M}^{K^*}_{\Delta\lambda_\mu}\right|^2,$$

zfit - Feedback From Users

- Primary goal: Have an accessible pythonic fitting framework, beneficial for both new and experienced users
- Integrated well within the general Python data science ecosystem in general and Scikit-HEP in particular
- Continuing to further pursue integration for plotting routines and different statistical treatments (some with hepstats)
- Some PDFs not supported $\rightarrow$ potential for users to fallback to RooFit
- The package would benefit greatly from additional support and feedback from stats experts! $\rightarrow$ Fitting is a huge topic, with many avenues. Need support from experts in the implementation of specific statistic recipes, either in development or feedback form

PIDCalib2 - ROOT-less Particle ID Efficiency Estimation

Software tools are needed to extract PID calibration between data and MC

uproot and boost-histogram:

- Simple access to ROOT files with Python, and subsequent histograms
- Allows higher-dimensional binning

matplotlib and mplhep

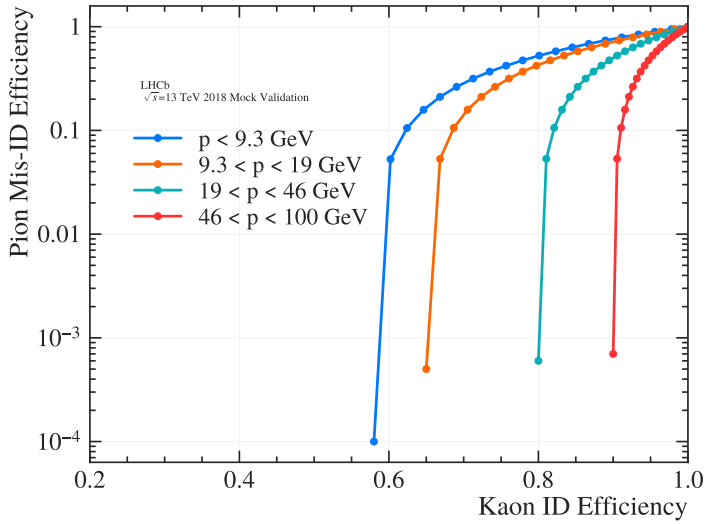
- Simple plotting, easy installation, lightweight
- mplhep to apply LHCb-specific style

xrootd

- Accessing remote ROOT files with a pythonic interface

pandas

- Manipulating tabular ROOT data to vectorize efficiency calibration.
- Significant improvement over previous C++ iteration, primarily from installation



PIDCalib2 - Feedback From Users

- Python packages are simple to install and lightweight $\rightarrow$ great to ship to analysers to use the end product.
- Using large data formats for centralized calibrations can become costly $\rightarrow$ Looking for ways to cache between users to lower overhead
- Event-by-event calculation is cumbersome $\rightarrow$ huge benefit to vectorisation. Need to support new users by sharing examples of how vectorization can be implemented in their code.
- Documentation is very good for some Python HEP packages, and very poor for others $\rightarrow$ installation and implementation headaches don't outweigh the benefits of the packages

Additional Tools Utilized Within LHCb

Highlighting some additional tools being utilized (and some developed!) by LHCb collaborators:

Particle

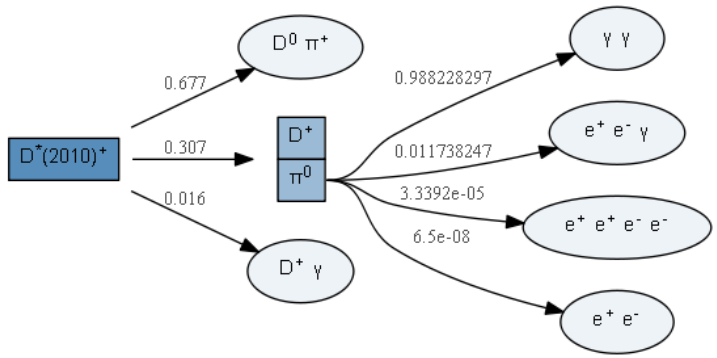
- Particle is a simple pythonic interface for PDG information.
- Extremely handy for analyzers with many decays to consider in a single analysis.
  - Some analyses utilize over 100 individual PDG objects!

pyhf

- New rare-decay searches benefit from the simple implementation to an ecosystem that does not use much native HistFactory/HistFitter.

DecayLanguage

- Visualizing and manipulating decay chains with a Python tool.
- Often taking the .dec files (basis of LHCb MC generation) as inputs, users can produce simple, but informative, infographic describing their particle decays



Feedback from LHCb Developers and Users

We are developers:

- LHCb members play a big part in driving the developments of the HEP software ecosystem on many fronts → Not just users!
- Developing the tools in house allows them to be directly applied to analyses and other LHCb-specific software tasks → lots of room for more hands to generalize!
- "Standard tools" draw more interest → more application to life beyond academia
- Students often learning Python tools in university classes → attraction to Python tools continues to grow
- Striving for outreach and education to get packages into the hands of new users

As users:

- Very important that tools have relevant documentation → Huge advantage to accessibility if working examples exist as well!
- Availability of testing tools → analyses have lots of specific use cases, test beds for a wide scope is valuable, especially for younger collaborators!
- Bug fixes implemented quickly!
- Installation is quick and simple → huge bonus over ROOT software
- New users can be overwhelmed with choice and struggle to know which tools to use → increased training (like this workshop!) and improved documentation can help users to make informed choices

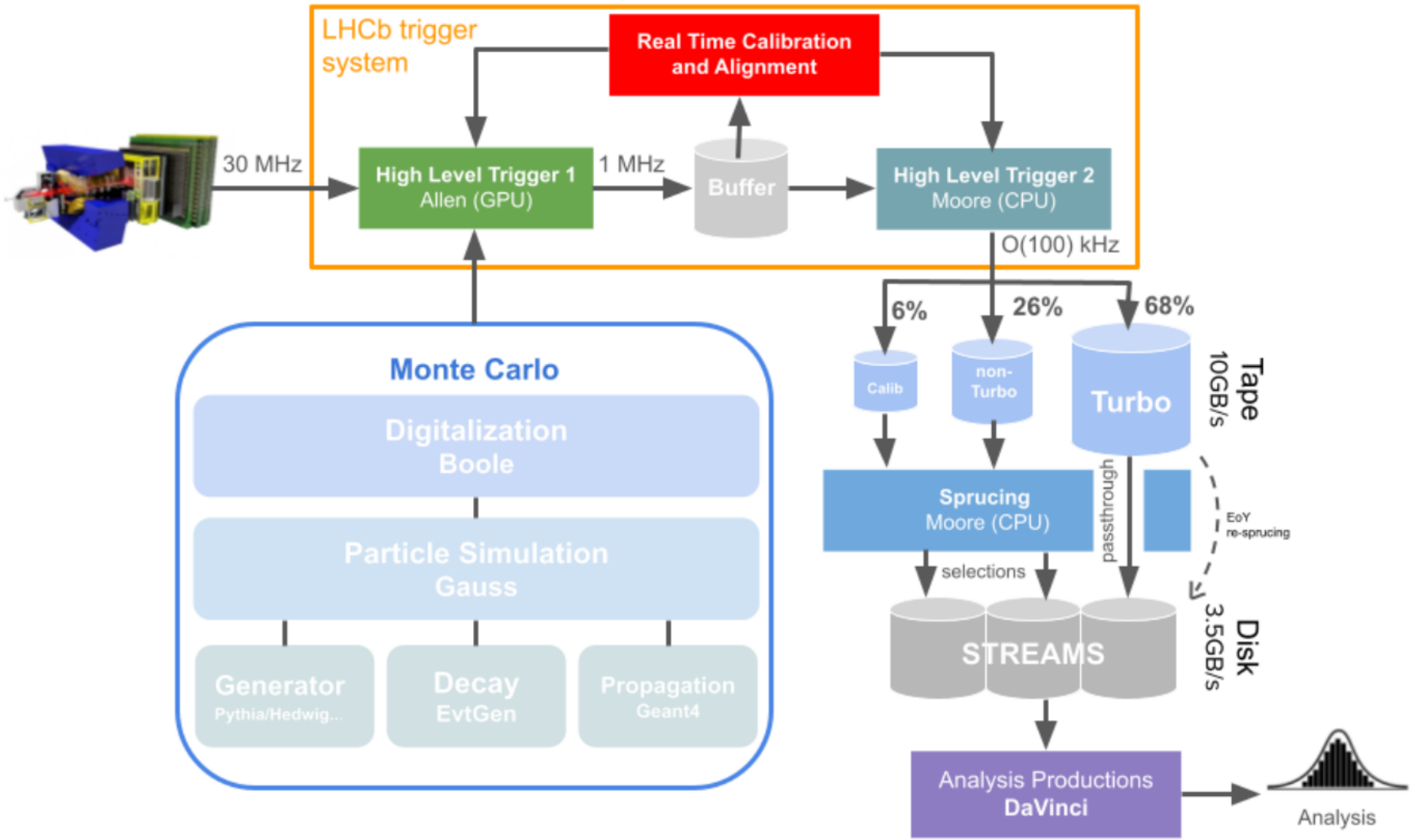
A lot of advantage to using Scikit-HEP and Python HEP tools

→ Keep them coming!

Discussion

BACKUP

Full data workflow



▸ LHCB-FIGURE-2020-016