Snakemake in LHCb

Benjamin Couturier LHCb Software and Computing Board chair

LHCb collaboration

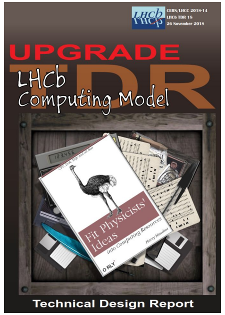
~1500 collaborators

~20 countries

100 institutes

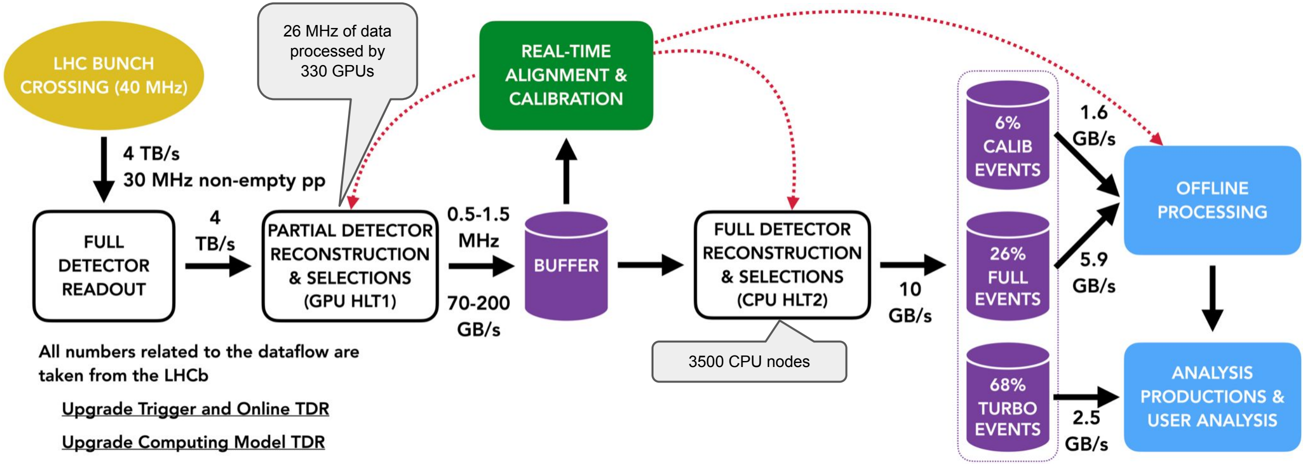
From https://cds.cern.ch/record/2319756/files/LHCB-TDR-018.pdf

LHCb computing model



Full detector readout with Software High-Level-Trigger (HLT1 on GPU, HLT2 on CPU) producing 10 GB/s of data from HLT2, split in 3 different streams:

- **Turbo** events contain only the part of the event relevant to the physics analysis
- Calibration (**Turcal**) streams with the data needed to study detector/tracking/pid performance
- **FULL** events, restricted to some analyses that cannot be accommodate with the turbo model


LHC BUNCH CROSSING (40 MHz)

4 TB/s
30 MHz non-empty pp

26 MHz of data processed by 330 GPUs

FULL DETECTOR READOUT

4 TB/s

PARTIAL DETECTOR RECONSTRUCTION & SELECTIONS (GPU HLT1)

0.5-1.5 MHz

70-200 GB/s

BUFFER

REAL-TIME ALIGNMENT & CALIBRATION

FULL DETECTOR RECONSTRUCTION & SELECTIONS (CPU HLT2)

3500 CPU nodes

10 GB/s

6% CALIB EVENTS

26% FULL EVENTS

68% TURBO EVENTS

1.6 GB/s

5.9 GB/s

2.5 GB/s

OFFLINE PROCESSING

ANALYSIS PRODUCTIONS & USER ANALYSIS

All numbers related to the dataflow are taken from the LHCb

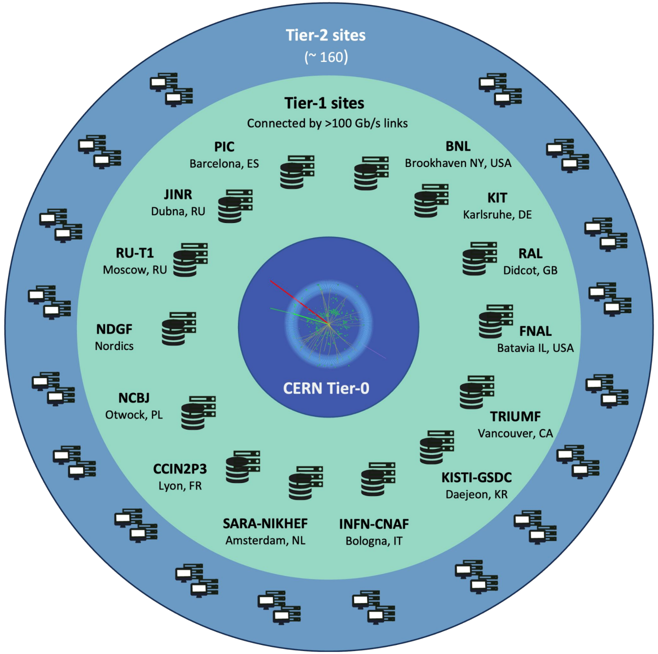
Upgrade Trigger and Online TDR

Upgrade Computing Model TDR

LHCb Distributed computing and WLCG

DIRAC
THE INTERWARE

LHCb uses DIRAC to process its data on the WLCG, with ~120 PB of data stored on tape so far and ~100k jobs running at any time (95% of the CPU load being simulation).



WLCG
Worldwide LHC Computing Grid

see here

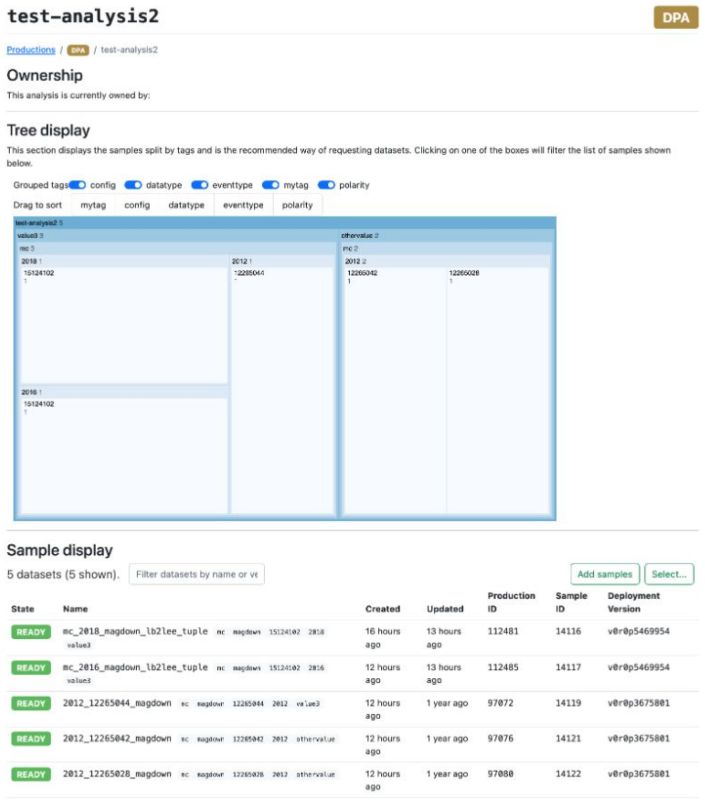
LHCb Analysis model

Facing the amount to be analyzed in Run3, LHCb put in place the Analysis Production system, to produce **ntuples** on the grid:

- in a **reproducible** way ⇒ *git/gitlab used to track the code*
- in a more **reliable** way ⇒ continuous integration used to test the requests on one file before submitting the request thus limiting further job failures
- in a way that **encourages reuse**, as the samples produced are listed in a common database, and can be tagged by analysts to indicate their purpose

The whole offline chain run this year, producing ntuples **within 3 days** of data being processed by HLT2

*Currently ~**100 Analysis Productions** are "live" and pick up data automatically*



test-analysis2

Productions / DPA / test-analysis2

Ownership

This analysis is currently owned by:

Tree display

This section displays the samples split by tags and is the recommended way of requesting datasets. Clicking on one of the boxes will filter the list of samples shown below.

Sample display

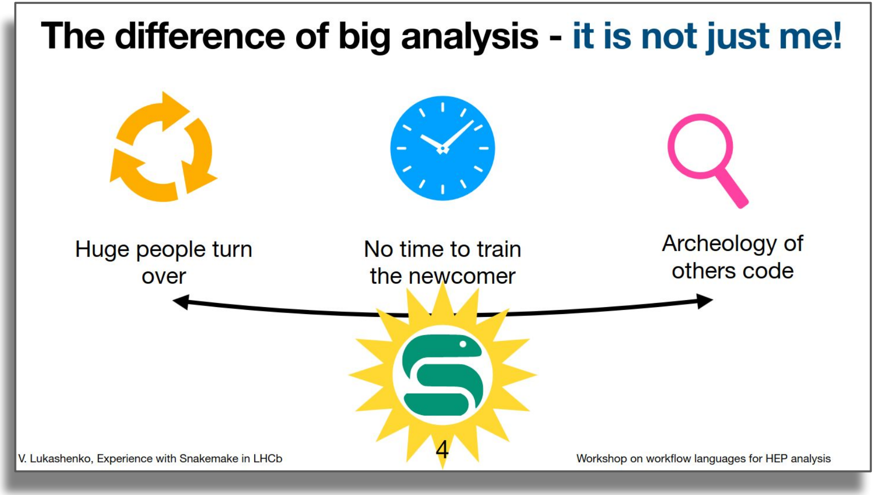
5 datasets (5 shown).

State	Name	Created	Updated	Production ID	Sample ID	Deployment Version
READY	mc_2018_magdown_lb2lee_tuple mc magdown 15124102 2018 value3	16 hours ago	13 hours ago	112481	14116	v0r0p5469954
READY	mc_2016_magdown_lb2lee_tuple mc magdown 15124102 2016 value3	12 hours ago	13 hours ago	112485	14117	v0r0p5469954
READY	2012_12265044_magdown mc magdown 12265044 2012 value3	12 hours ago	1 year ago	97072	14119	v0r0p3675801
READY	2012_12265042_magdown mc magdown 12265042 2012 othervalue	12 hours ago	1 year ago	97076	14121	v0r0p3675801
READY	2012_12265028_magdown mc magdown 12265028 2012 othervalue	12 hours ago	1 year ago	97080	14122	v0r0p3675801

Challenges of data analysis

Analyses take several years and involve (changing) teams of analysts

The need to structure analysis code was presented very clearly by V.Lukashenko at the Workshop on workflow languages for HEP analysis (April 2024):



The difference of big analysis - it is not just me!

Huge people turn over

No time to train the newcomer

Archeology of others code

V. Lukashenko, Experience with Snakemake in LHCb

4

Workshop on workflow languages for HEP analysis

Snakemake can help, as it is a very analyst friendly tool and it eases workflow preservation.

Snakemake benefits

Workflow tools and Snakemake in particular can help structuring analysis code (C.f. V.Lukashenko's talk)

- Workflows defined in a Python based language, easing development
- Workflows can be re-run automatically
- Easy looping over parameter spaces
- Snakemake allows for easy parallelization of the processing

And also:

- Helps keep track of dependencies between artefacts derived from LHCb data
- Integration with the XRootD protocol is crucial for LHCb
- Continuous integration systems can be used to rerun workflows

Practical use of Snakemake

It’s not always so easy!

- Snakemake needs to be available in the environment, with the right plugins
- authentication/authorization for data access (especially from CI workflows)
- Need to keep track of the link with production data
- Workflows can be overcomplicated (and difficult to understand)

So:

⇒ LHCb default analysis environment (using conda) provides a Snakemake installation

⇒ the CERNVM File System is key for (efficient) software deployment in LHCb
(we do not distribute full containers)

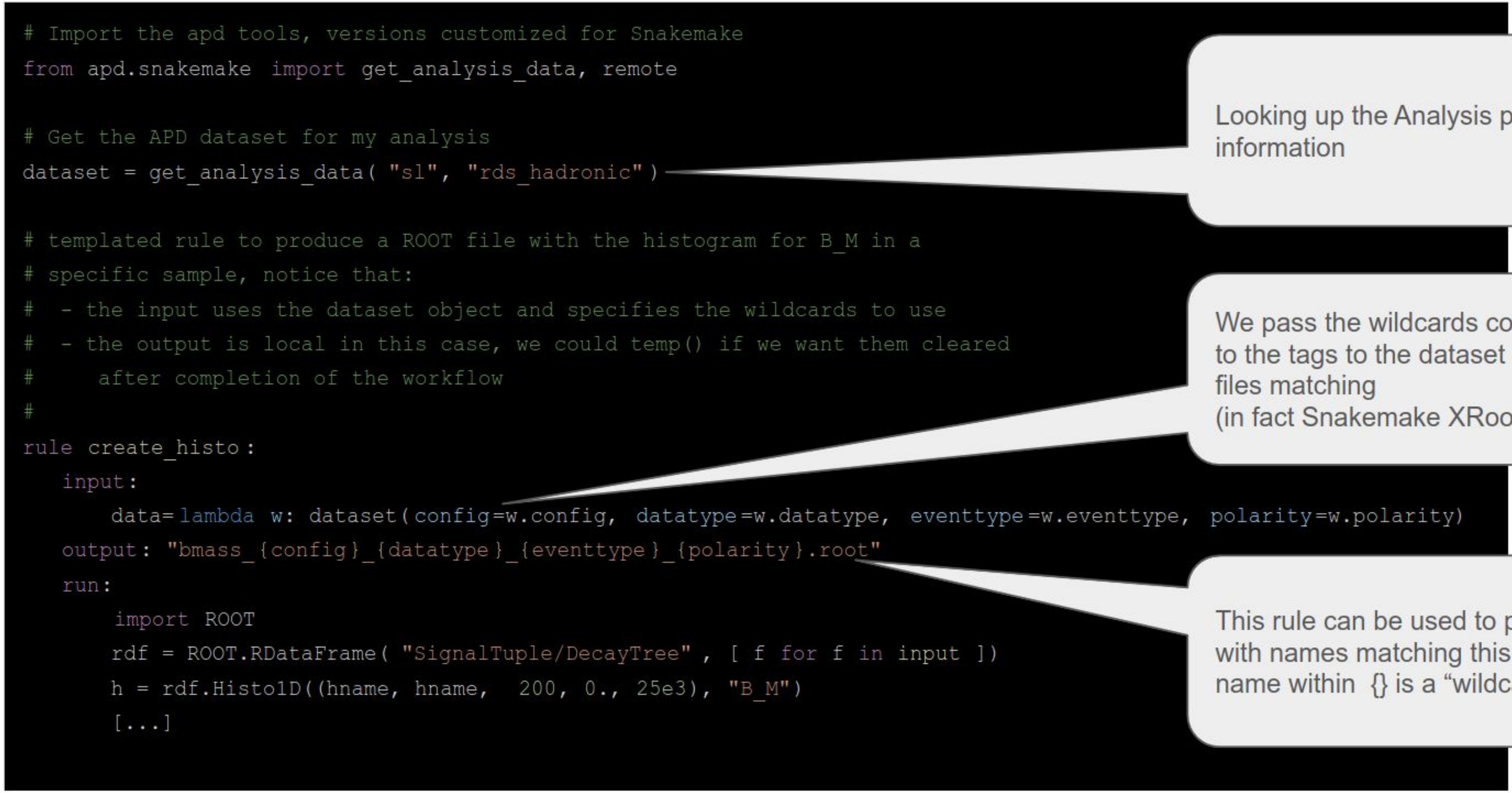
Data integration

⇒ Developed the Analysis Production Data (`apd`) python package to easily query the LHCb bookkeeping database

⇒ It can also be used to get EOS tokens to give access to data

⇒ Snakemake interface to `apd` also developed

Data integration



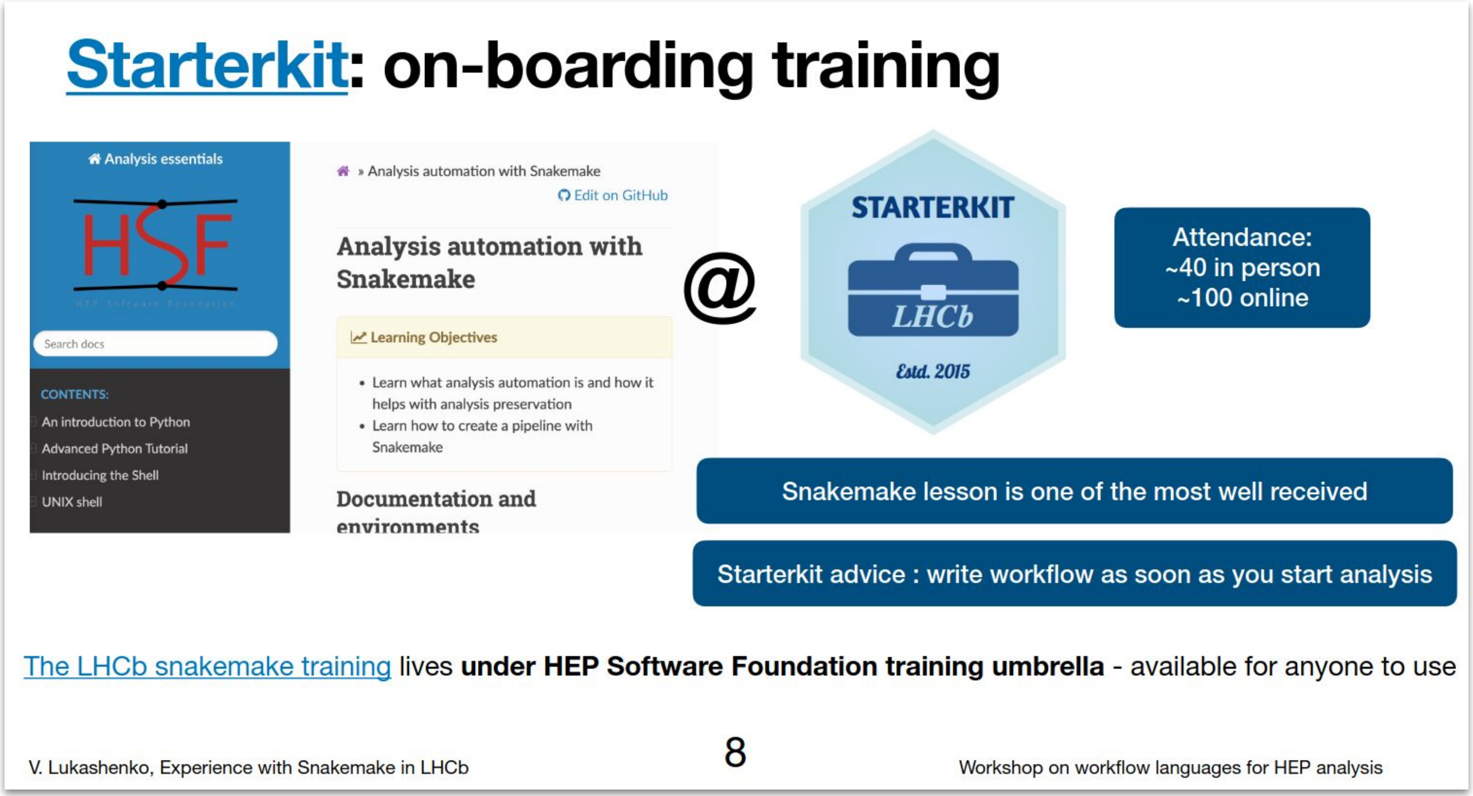
```
# Import the apd tools, versions customized for Snakemake
from apd.snakemake import get_analysis_data, remote

# Get the APD dataset for my analysis
dataset = get_analysis_data( "sl", "rds_hadronic" )

# templated rule to produce a ROOT file with the histogram for B_M in a
# specific sample, notice that:
#  - the input uses the dataset object and specifies the wildcards to use
#  - the output is local in this case, we could temp() if we want them cleared
#     after completion of the workflow
#
rule create_histo :
   input:
       data=lambda w: dataset(config=w.config, datatype=w.datatype, eventtype=w.eventtype, polarity=w.polarity)
   output: "bmass_{config}_{datatype}_{eventtype}_{polarity}.root"
   run:
       import ROOT
       rdf = ROOT.RDataFrame( "SignalTuple/DecayTree" , [ f for f in input ])
       h = rdf.Histo1D((hname, hname,  200, 0., 25e3), "B_M")
       [...]
```

Looking up the Analysis production information

We pass the wildcards corresponding to the tags to the dataset to get a list of files matching
(in fact Snakemake XRootD remotes)

This rule can be used to produce files with names matching this form. Each name within {} is a “wildcard”

Training is key



Starterkit: on-boarding training

Analysis essentials

Search docs

CONTENTS:

- An introduction to Python
- Advanced Python Tutorial
- Introducing the Shell
- UNIX shell

» Analysis automation with Snakemake

Edit on GitHub

Analysis automation with Snakemake

Learning Objectives

- Learn what analysis automation is and how it helps with analysis preservation
- Learn how to create a pipeline with Snakemake

Documentation and environments

@

Attendance:
~40 in person
~100 online

Snakemake lesson is one of the most well received

Starterkit advice : write workflow as soon as you start analysis

The LHCb snakemake training lives **under HEP Software Foundation training umbrella** - available for anyone to use

V. Lukashenko, Experience with Snakemake in LHCb 8 Workshop on workflow languages for HEP analysis

Snakemake and HTCondor

Running on batch systems such as HTCondor is possible, workflows making the parallelization easy, see:

- V.Lukashenko ‘s talk at the HEP Software Data Analysis Workgroup
- J.Gooding’s talk at FOSDEM 24

In practice need to see about data access, authentication and authorization etc…

Outlook

Snakemake is a very useful tool to organize the LHCb data analyses, used by some analysts in LHCb for a long time.

Efforts are made to ease its use by LHCb physicists:

- Training
- Integration with the bookkeeping
- Data access via the XRootD

But some more efforts are needed (e.g. XRootD integration in Snakemake >= 8)