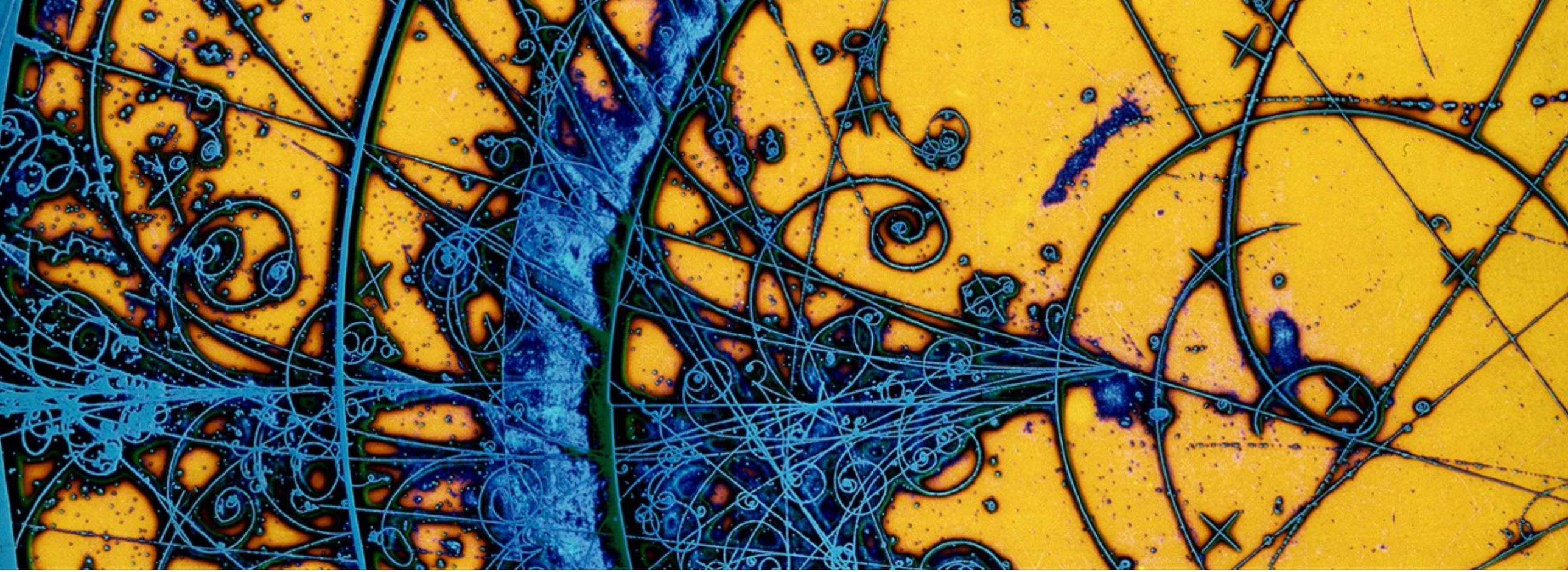
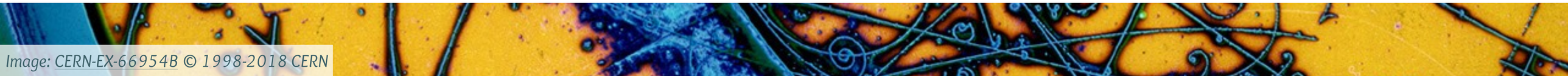


Analysis Productions: A declarative approach to ntupling

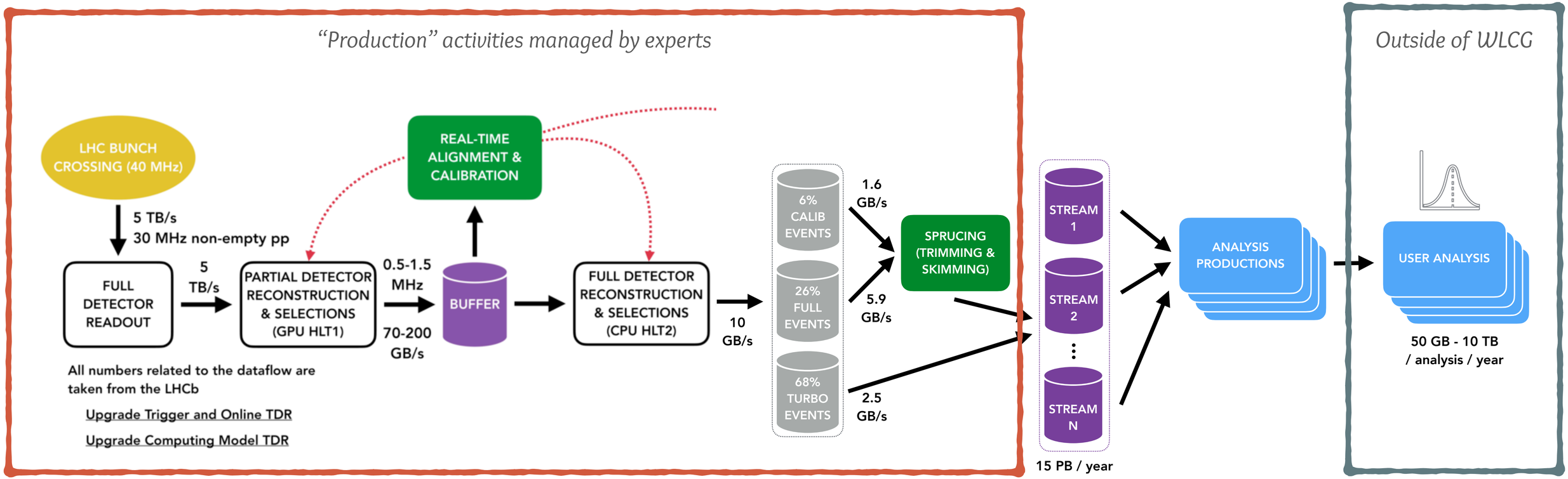
Chris Burr on behalf of the LHCb collaboration

9th May 2023



LHCb data processing

- Everything before "user analysis" is **embarrassingly parallel**
- Almost all LHCb analyses start by making ROOT TTrees using the LHCb physics applications
  - Filter large datasets down to something manageable with local resources
  - Later analysis is done outside of the LHCb software (ROOT/pandas/numpy/...)
- Historically analysts were responsible for running O(10,000) grid jobs to produce ROOT files



Issues with the “user jobs” model

- It’s really time consuming
  - Everyone doing analysis needs to learn how the grid works and how to handle transient errors
- There is a provenance gap between “production” and “user analysis”
  - Documenting how data is produced is left to analysts
  - Lack of visibility and data reuse
    - Emails and copying data to public locations to avoid reprocessing
- Instability
  - Did you test that before submitting?

Fundamentally* the activity is the same as “production” activities

What does it mean to produce files?

1. Pass some configuration to an application that converts input file -> output file
2. Run that application on all the files that contain physically-similar data (or simulation)
3. Repeat for all the different types of input you're interested (ranges from O(1) → O(1000)*)

```yaml
MySample:
  wg: WG
  inform:
    - someone@cern.ch

  application: DaVinci/v45r4
  options:
    - make_ntuple.py
  output: DVNtuple.root

  input:
    bk_query: /some/bookkeeping/path/
```

1. What to do

2. What input data

Also support chaining productions, e.g.
Run HLT1 then
use as input for HLT2 then
as input for ntupling

** 2 magnet polarities × 6 years × (data + simulation) = 24 productions for a simple† analysis*

† Common to have many final states and many simulation samples per final state

Repeating

```yaml
defaults:
  application: DaVinci/v45r4
  wg: WG
  automatically_configure: yes
  turbo: no
  inform:
    - someone@cern.ch
  options:
    - make_ntuple.py
  output: DVNtuple.root

{%- set datasets = [
  (11, 3500, '14', '21r1'),
  (12, 4000, '14', '21'),
  (15, 6500, '15a', '24r2'),
  (16, 6500, '16', '28r2'),
  (17, 6500, '17', '29r2'),
  (18, 6500, '18', '34'),
]%}

{%- for year, energy, reco, strip in datasets %}
  {%- for polarity in ['MagDown', 'MagUp'] %}

My_20{{year}}_{{polarity}}_job:
  input:
    bk_query: /LHCb/Collision{{year}}/Beam{{energy}}GeV-VeloClosed-{{polarity}}/Real Data/Reco{{reco}}/Str

  {%- endfor %}
{%- endfor %}
```

Use Jinja templating
to “render” the YAML

Getting it into production

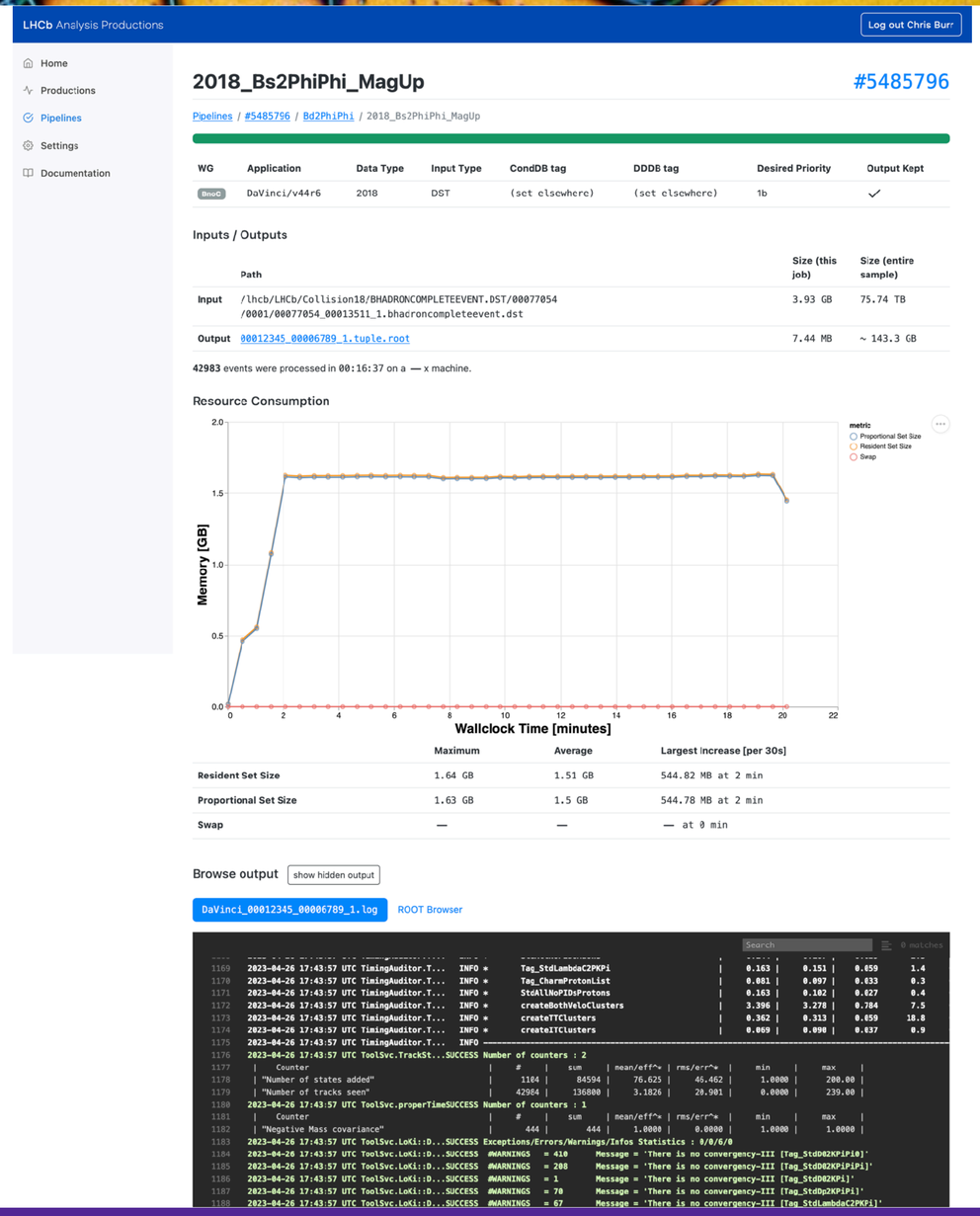
Fundamentally* the activity is the same as “production” activities

** There are more different configurations used by analysts*
** User jobs have much higher failure rates*

- LHCb grid operations consists of very few people
  - Can’t have buggy user productions causing chaos
- Use CI/CD to submit productions? Similar but some differences
  - Most CI systems run the same pipeline repeatedly
  - Spikey compute requirements, often need to test O(1000) different productions
  - Not really pass/fail, often want to closely examine the output of the test to see if it’s “reasonable”

Viewing CI results

- Generally: If one file works, 99%+ of files will
- Submit each test as a separate grid job
  - Can easily handle the spiky demand
  - Output streamed to a web application for quick feedback
- Each test is displayed on a dedicated page
  - View logs
  - Monitor resource usage with prmon
  - Download output data
  - Browse output data with JSROOT
  - Can include post-validation of output data “checks”



Checks

State	Check	Trees	Messages
PASS	turbo_MagUp_2022data_Tuple / B_mass	BsToVG/DecayTree	Histogram of B_M successfully filled from TTree BsToVG/DecayTree (contains 30.0 events)
		BdToVG/DecayTree	Histogram of B_M successfully filled from TTree BdToVG/DecayTree (contains 37.0 events)
		BdToHHG/DecayTree	Histogram of B_M successfully filled from TTree BdToHHG/DecayTree (contains 7.0 events)
		BsToHHG/DecayTree	Histogram of B_M successfully filled from TTree BsToHHG/DecayTree (contains 7.0 events)

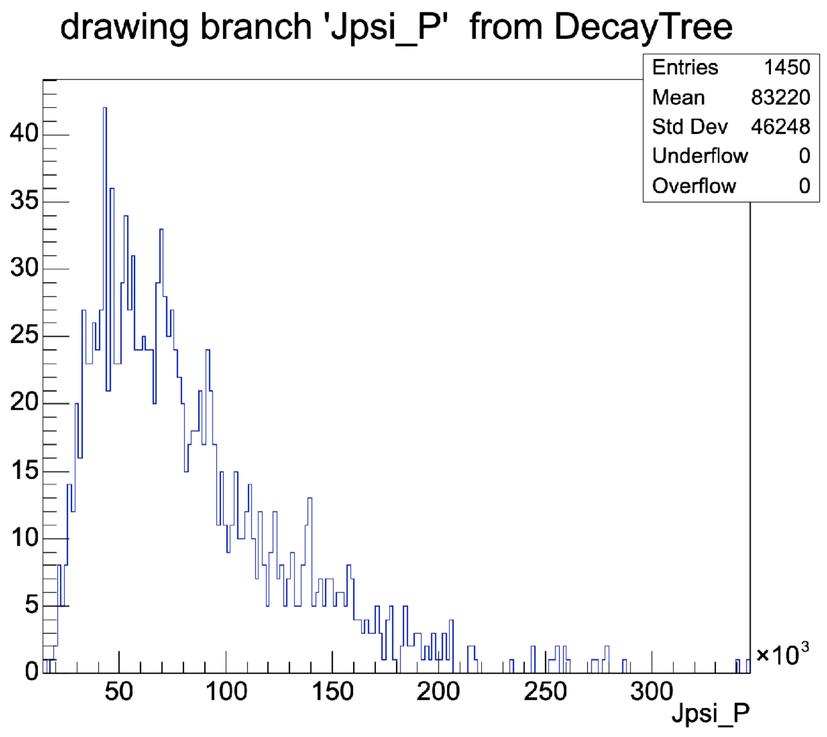
Viewing CI results

DaVinci_00012345_00006789_1.log | ROOT Browser

Read a ROOT file

JSROOT version 7.2.0 11/08/2022

- Bp_Hlt1_Hlt1TrackMuonMVA
- Bp_Hlt1_Hlt1TwoTrackMVADe
- Bp_Hlt1_Hlt1TwoTrackMVADe
- Jpsi_ID
- Jpsi_KEY
- Jpsi_PT
- Jpsi_SUM_PT
- Jpsi_MIN_PT
- Jpsi_MAX_PT
- Jpsi_PX
- Jpsi_PY
- Jpsi_PZ
- Jpsi_E
- Jpsi_P
- Jpsi_SUM_P
- Jpsi_MIN_P
- Jpsi_MAX_P
- Jpsi_M
- Jpsi_FOURMOMENTUME
- Jpsi_FOURMOMENTUMX
- Jpsi_FOURMOMENTUMY
- Jpsi_FOURMOMENTUMZ
- Jpsi_BPVDIRA
- Jpsi_BPVFD
- Jpsi_BPVFDCHI2
- Jpsi_BPVIPCHI2
- Jpsi_MINIPCHI2
- Jpsi_MINIP
- Jpsi_DOCA
- Jpsi_DOCACHI2
- Jpsi_CHI2
- Jpsi_CHI2VXNDOF
- Jpsi_BPVVDRHO
- Jpsi_BPVLTIME
- Jpsi_ETA
- Jpsi_PHI
- Jpsi_PVX
- Jpsi_PVY
- Jpsi_PVZ



JSROOT based browser

Log highlighting, search and filtering

Browse output | show less output

DIRAC.log | DaVinci_00012345_00006789_1.log | bookkeeping_00012345_00006789_1.xml | jobDescription.xml | pool_xml_catalog.xml

prmon.json | prmon.txt | prodConf_DaVinci_00012345_00006789_1.json | summaryDaVinci_00012345_00006789_1.xml | ROOT Browser

Counter | 16 matches

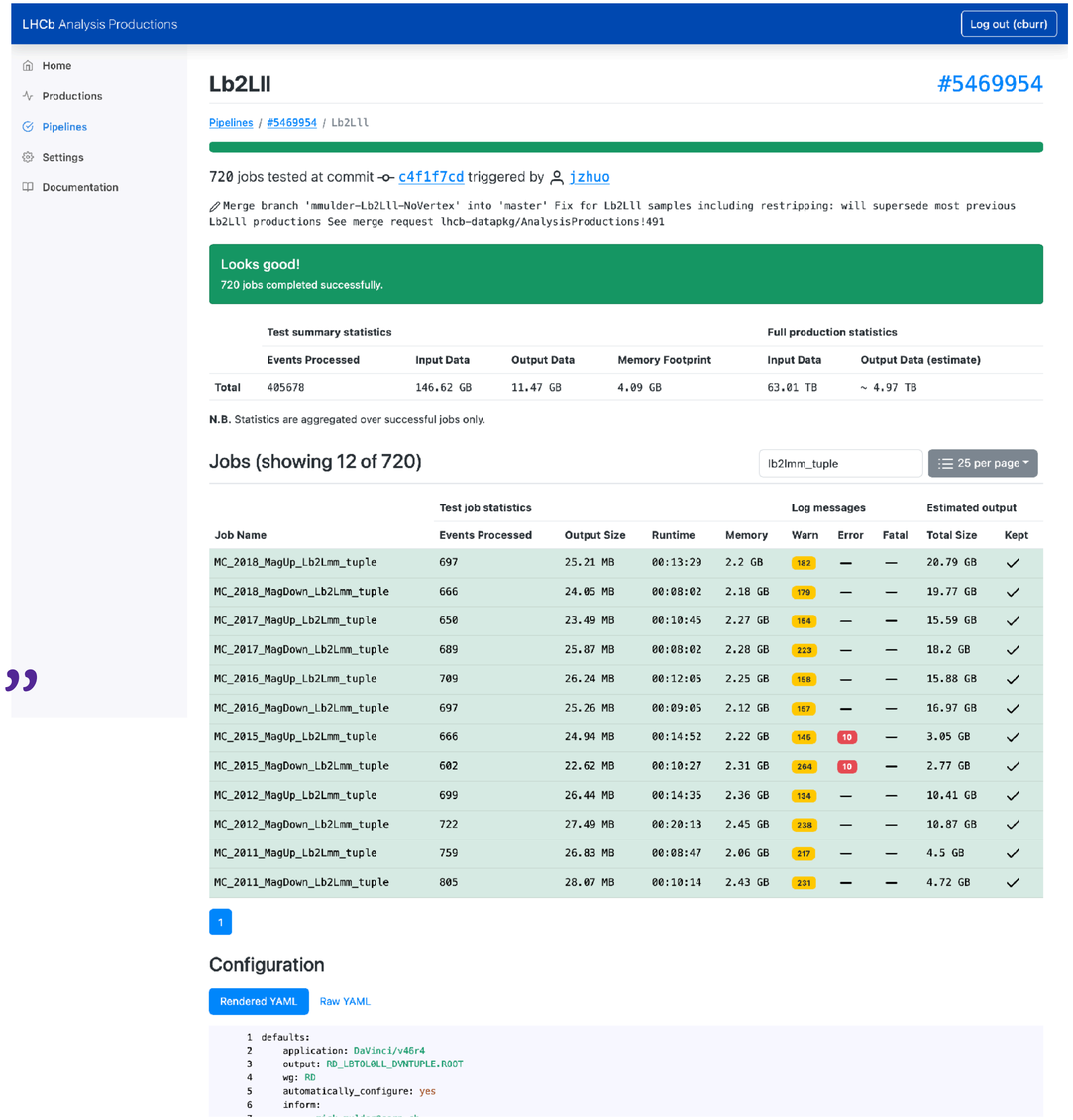
```
303 HDRFilter_BuToJpsiKplus_Detached      INFO Number of counters : 1
304  |    Counter                                 |    #    |    sum     | mean/eff^* | rms/err^*  |    min     |    max     |
306 ParticleUnpacker                      INFO Number of counters : 2
307  |    Counter                                 |    #    |    sum     | mean/eff^* | rms/err^*  |    min     |    max     |
310 RawBankSizeFilter                     INFO Number of counters : 2
311  |    Counter                                 |    #    |    sum     | mean/eff^* | rms/err^*  |    min     |    max     |
314 RecSummaryUnpacker                    INFO Number of counters : 1
315  |    Counter                                 |    #    |    sum     | mean/eff^* | rms/err^*  |    min     |    max     |
317 RecVertexUnpacker                     INFO Number of counters : 1
318  |    Counter                                 |    #    |    sum     | mean/eff^* | rms/err^*  |    min     |    max     |
320 ToolSvc.HltFactory                    INFO Number of counters : 1
321  |    Counter                                 |    #    |    sum     | mean/eff^* | rms/err^*  |    min     |    max     |
323 Tuple                                 INFO Number of counters : 12
324  |    Counter                                 |    #    |    sum     | mean/eff^* | rms/err^*  |    min     |    max     |
384 XMLSummarySvc                         INFO filling counters...
385 XMLSummarySvc                         INFO counters filled OK
```

Aggregating CI results

- Not practical to verify hundred of tests
- Summarise key statistics
  - Peak memory usage
  - Output data size
  - Warning/Error messages

"I often use the CI instead of local tests because it's so much easier to read"

- Associated CLI tooling
  - Run tests locally before committing
  - Interactively reproduce a CI test

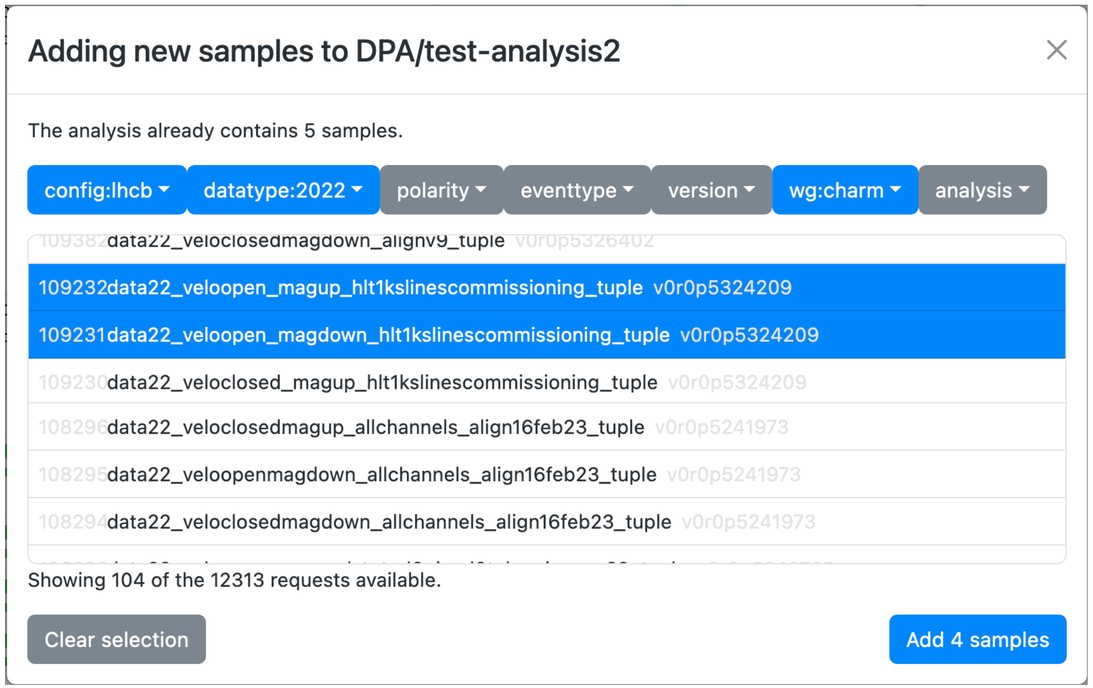


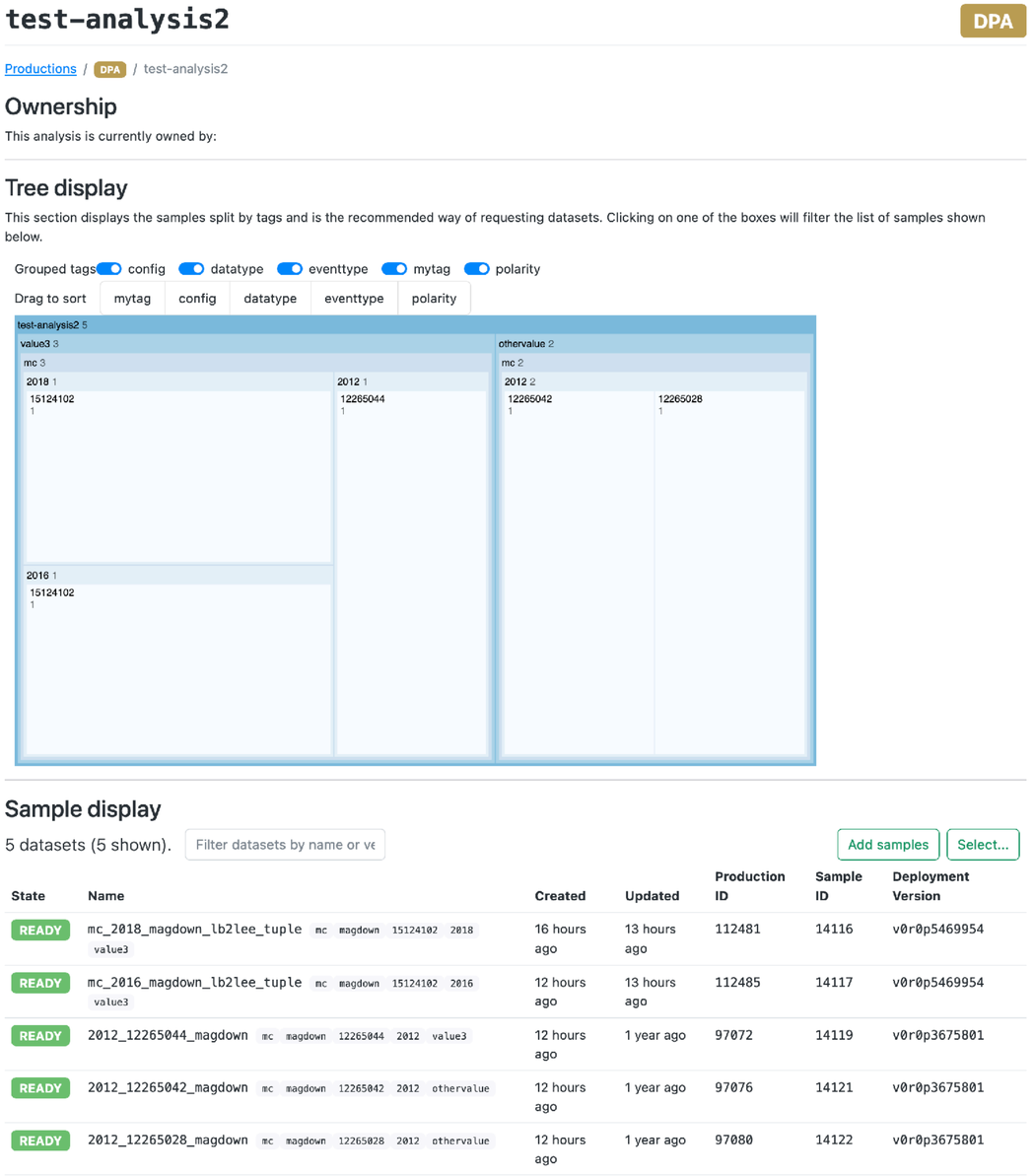
Submission

- Once the tests are passing, use a standard git merge request flow
  - Each “working group” has liasons responsible for reviewing/merging
  - Review is only for technical issues or identifying duplication
- Once merged a second CI pipeline starts:
  - The repository is tagged automatically
  - Dependencies are deployed to CVMFS
  - Productions are submitted
  - Open a ticket for tracking any processing problems that might arise
- The repository is automatically cleaned after each submission
  - No value keeping fails in the main branch - potentially misleading
  - Have tools for checking out past versions of productions for making revisions
- If no files are unprocessable, the process is fully automated

After the production starts

- Group data sets into an “analysis”
  - Automatically created when productions are submitted
  - Namespace split by area of the physics program
  - Each analysis is owned by a group of people
- Pre-existing samples can also be added





Accessing data

- We provide a tool named `apd` (analysis productions data)
- Look up data based on “meaningful tags” (year/magnet polarity/detector state/decay products)
- Provides provenance between the grid and local worlds
  - Simple interface which can provide local caching, authentication, long-term reproducibility
  - Well suited to analysis facilities
- All metadata is versioned with time to it’s possible to see the past state
- See “Facilitating the preservation of LHCb analyses” in Track 8

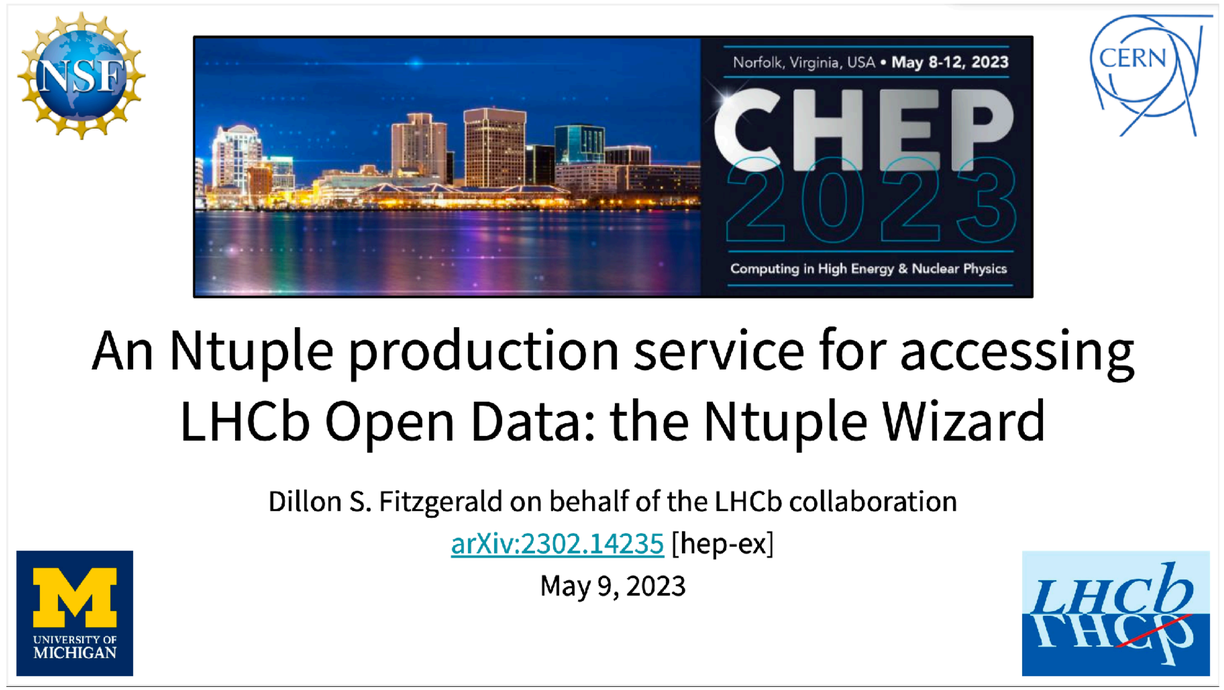
Unexpected benefits

- Review from liaisons is an opportunity for knowledge transfer
- Source of reference configurations
- Opportunities for static analysis
  - Both before and after data is processed

Extensions

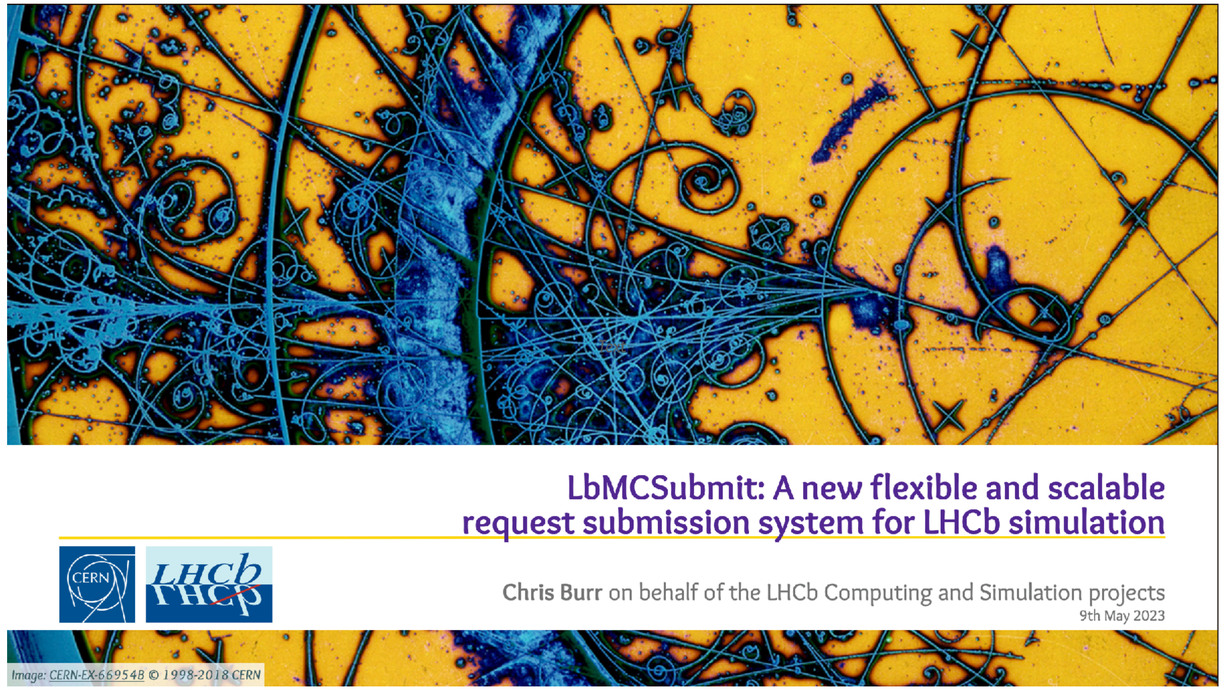
An Ntuple production service for accessing LHCb Open Data: the Ntuple Wizard

9th May 2023, 15:00



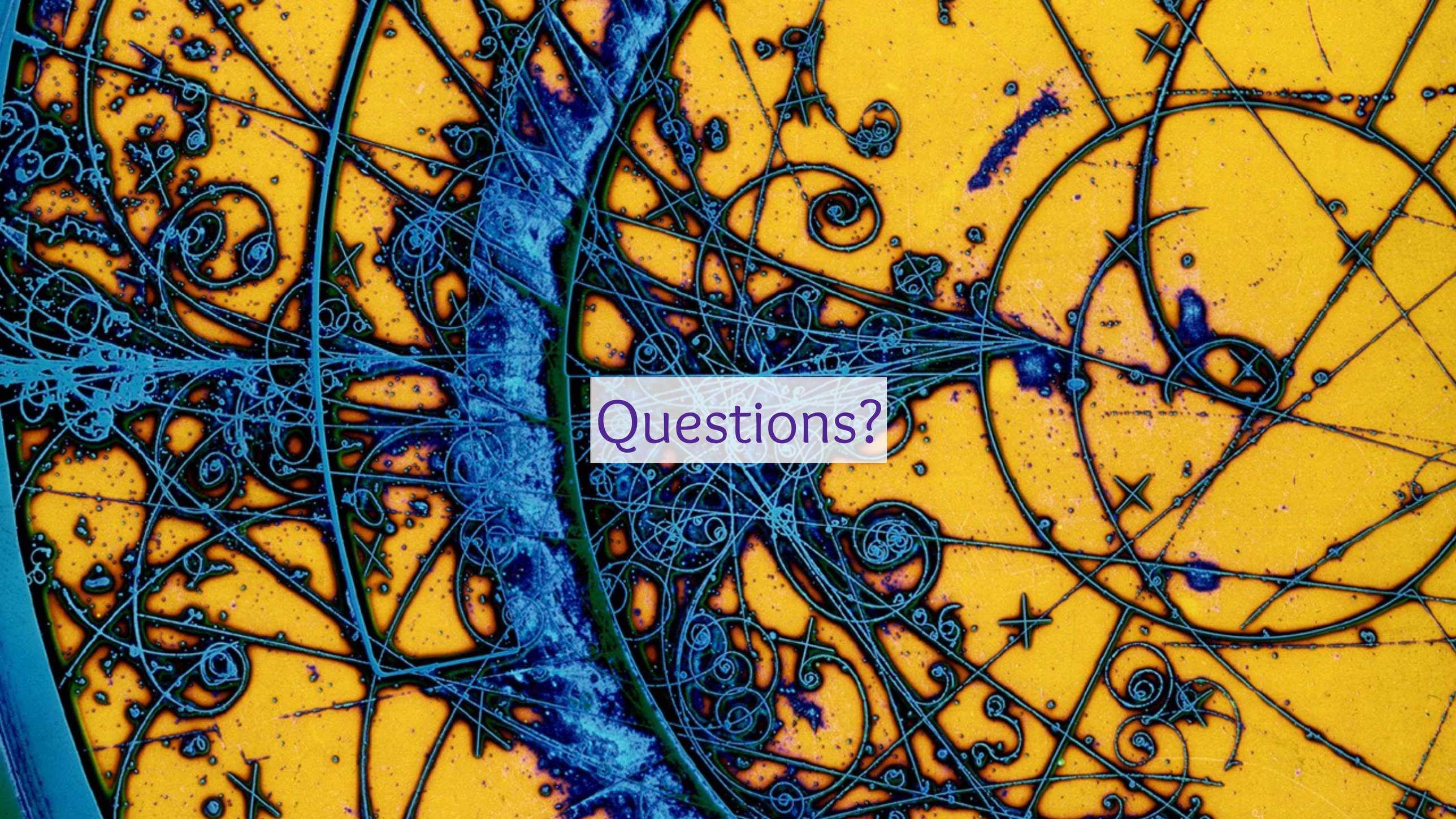
LbMCSubmit: A new flexible and scalable request submission system for LHCb simulation

9 May 2023, 17:15



Summary

- A new system for processing LHCb data has evolved over the last 5 years
- Has been used for running over 12,000 productions
  - For running over real and simulated data from LHC Run 1, 2 and 3
- Bridges the gap between grid and “user” worlds
  - Well suited to adapt to future changes (analysis facilities/protocols/authentication/...)


Questions?