Analysis Grand Challenge on REANA

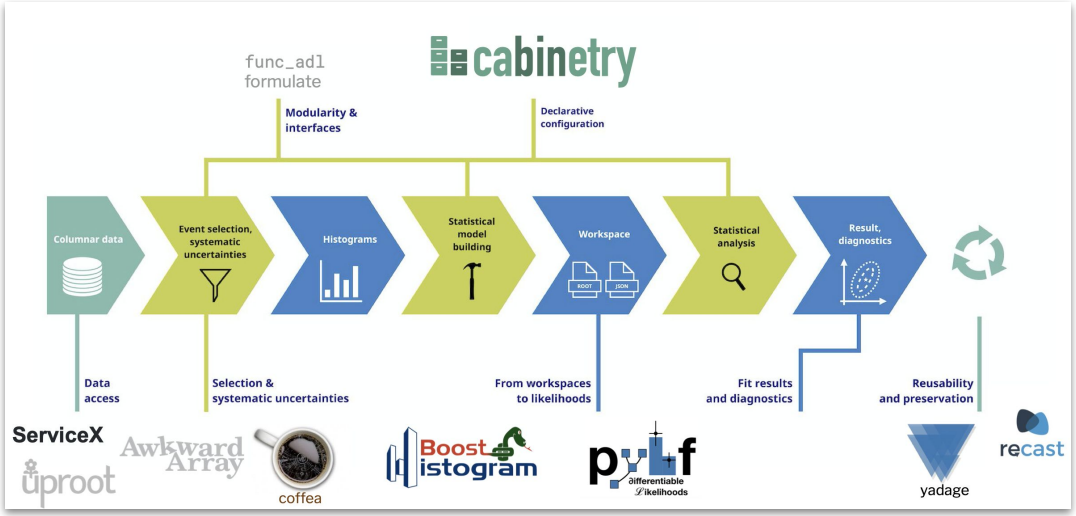
Andrii Povsten
Mentors: Alex Held, Matthew Feickert (University of Wisconsin- Madison), Oksana Shadura (University Nebraska-Lincoln), Tibor Simko (CERN)

FAIR and Open Science in High Energy Physics, OAC-2226378, OAC-2226379 and OAC-2226380

Analysis Grand Challenge IRIS-HEP implementation

- Columnar data extraction from large dataset
- Processing of that data (event filtering, construction of observables, evaluation of systematic uncertainties) into histograms
- Statistical model construction and statistical inference
- Relevant visualisation for this steps

\+ **Adding analysis preservation step to AGC pipeline**

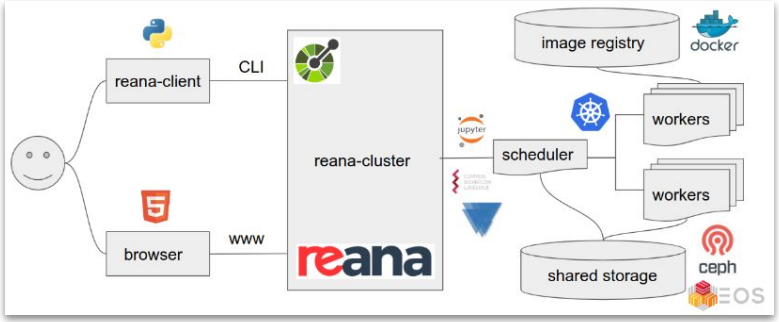


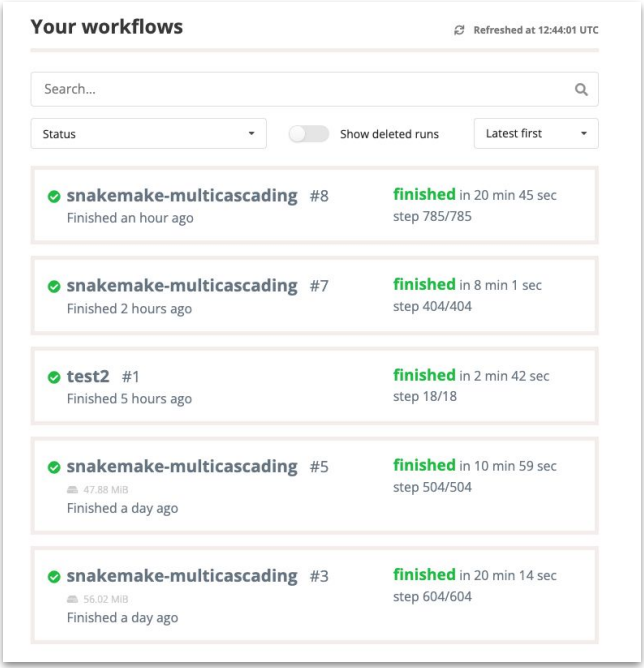
reana

HEPData

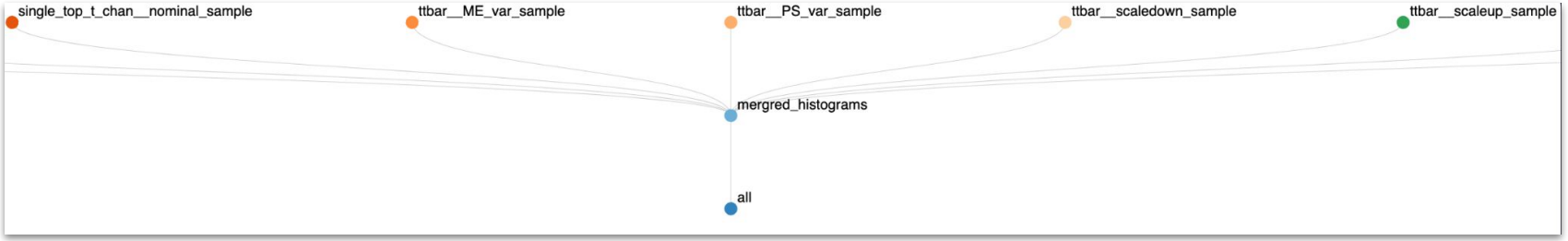
REANA - Reusable analysis

- Allow **complex multi-stage physics analysis to be executed with a single command, using REANA service**
- Enable **to submit parameterized computational workflows to run on remote compute clouds** or using other backends
- REANA **uses container technologies** to provide exact runtime environment necessary for various analysis steps
- Supports several **different container technologies** (Docker, Singularity), **compute clouds** (Kubernetes/OpenShift,), shared **storage systems** (Ceph, EOS) and **structured workflow specifications** (CWL, Yadage, Snakemake)





Analysis Grand Challenge pipeline: Adapting to Snakemake



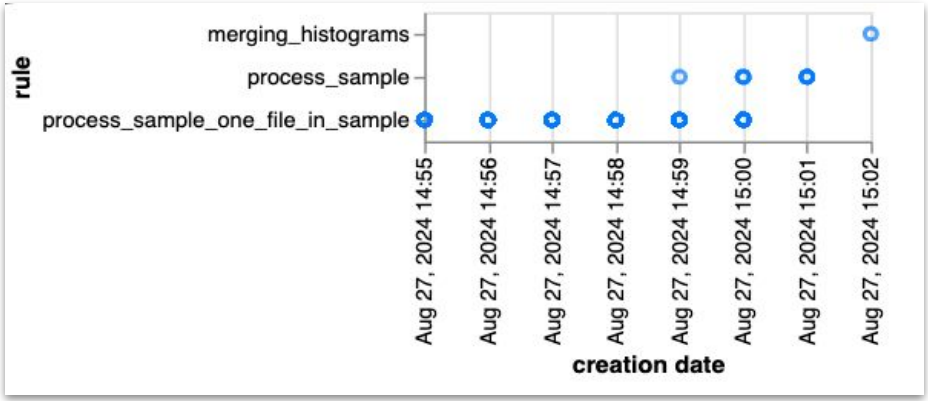
Each rule REANA sends to the Kubernetes cluster as separate node

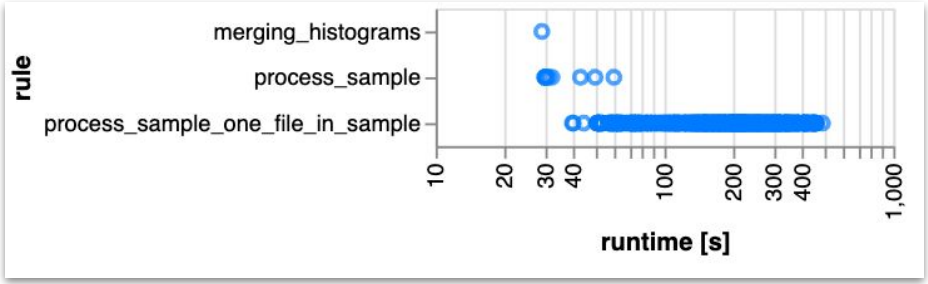
```
analyse file_ttbar_01 file_ttbar_02 file ...   file_wjets_01 file_wjets_02 file_wjets_03 ...

          \          |        /                        \         |          /
           \         |       /                          \        |         /

        merge sample ttbar_nominal                  merge sample wjets_nominal

                  \                                          /
                   \                                        /

                                merge all samples

                                       |
                                       |

                                     Plot
```

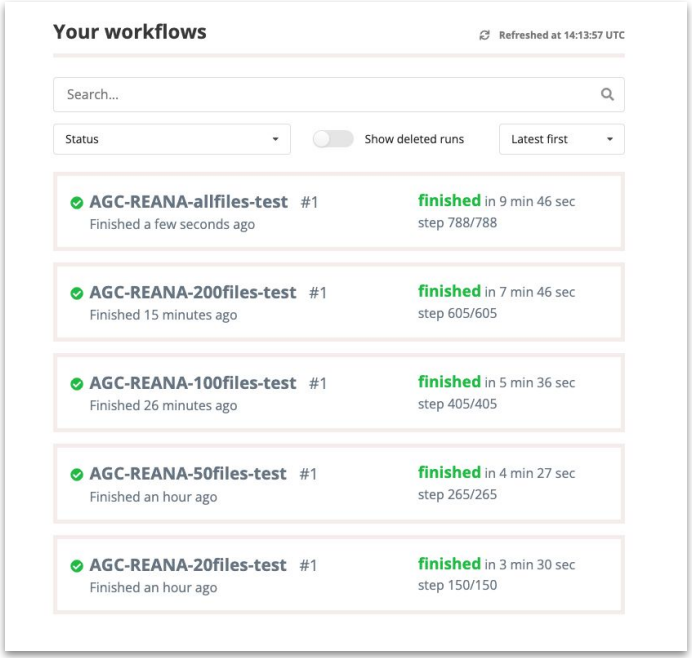
Snakemake checks the inputs and outputs in the rules to see the dependencies and order of execution

Achieved results:

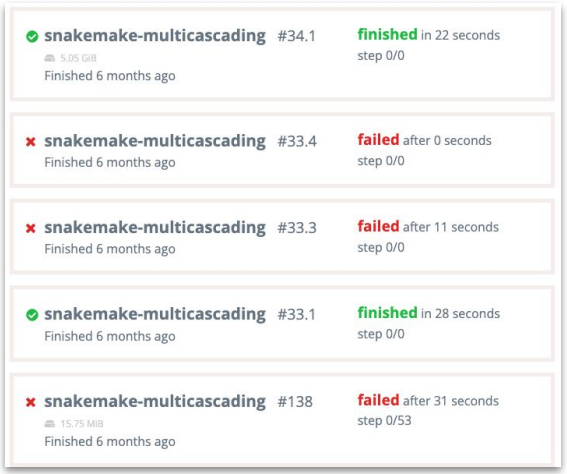
The processing and merging all files with final workspace.json file creation takes around 6-7 minutes with REANA.

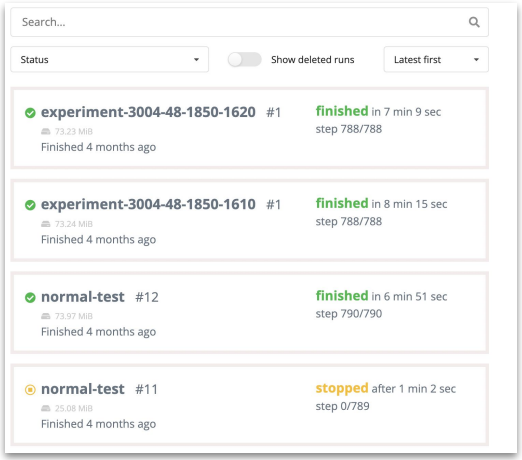


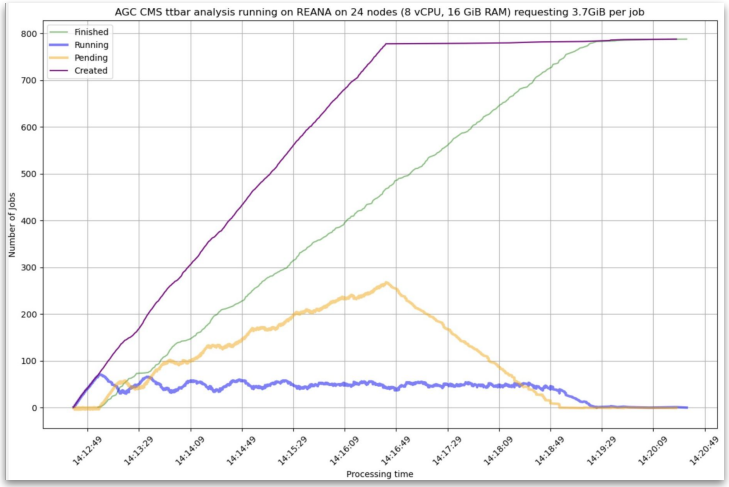


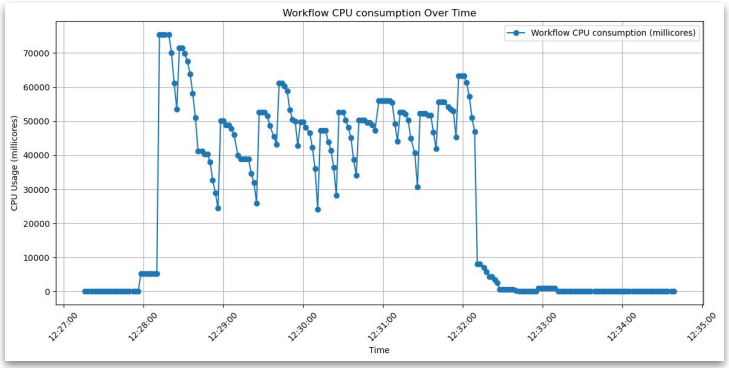


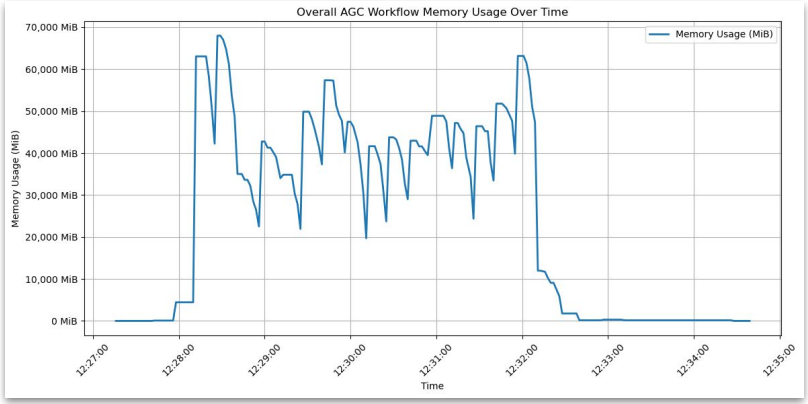
Run numerous benchmarking experiments







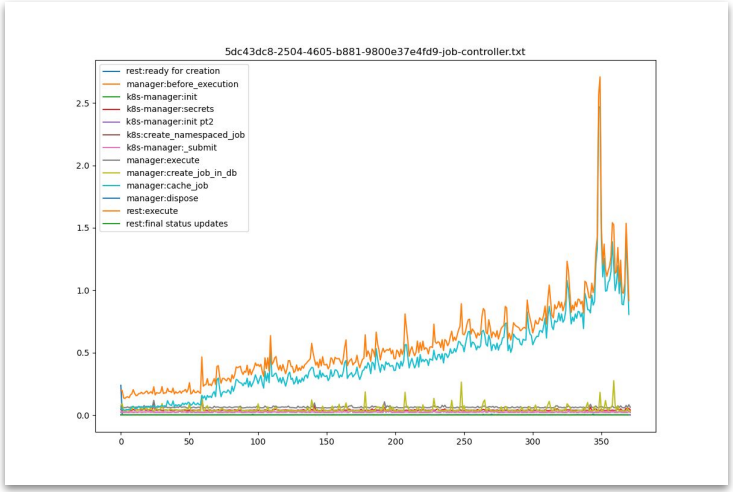


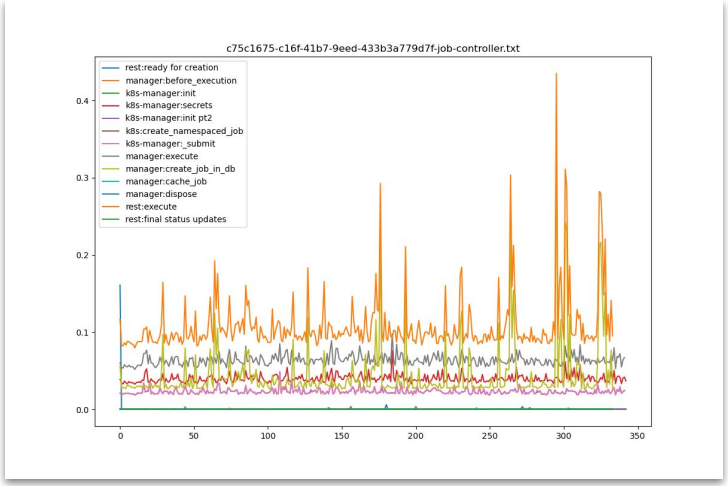


The AGC benchmarking experiments allowed to improve REANA performance:

- Snakemake engine version upgrade
- Avoid EOS input files throttling by using authenticated user
- Avoid unnecessary refetching of Kerberos secrets
- Faster terminating of jobs by improving refresh token sidecar termination

Cached improved mechanism example





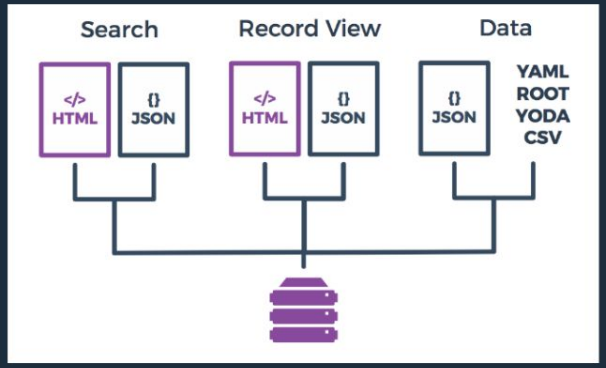
AGC and HEPData

HEPData

HEPData

Repository for publication-related High-Energy Physics data

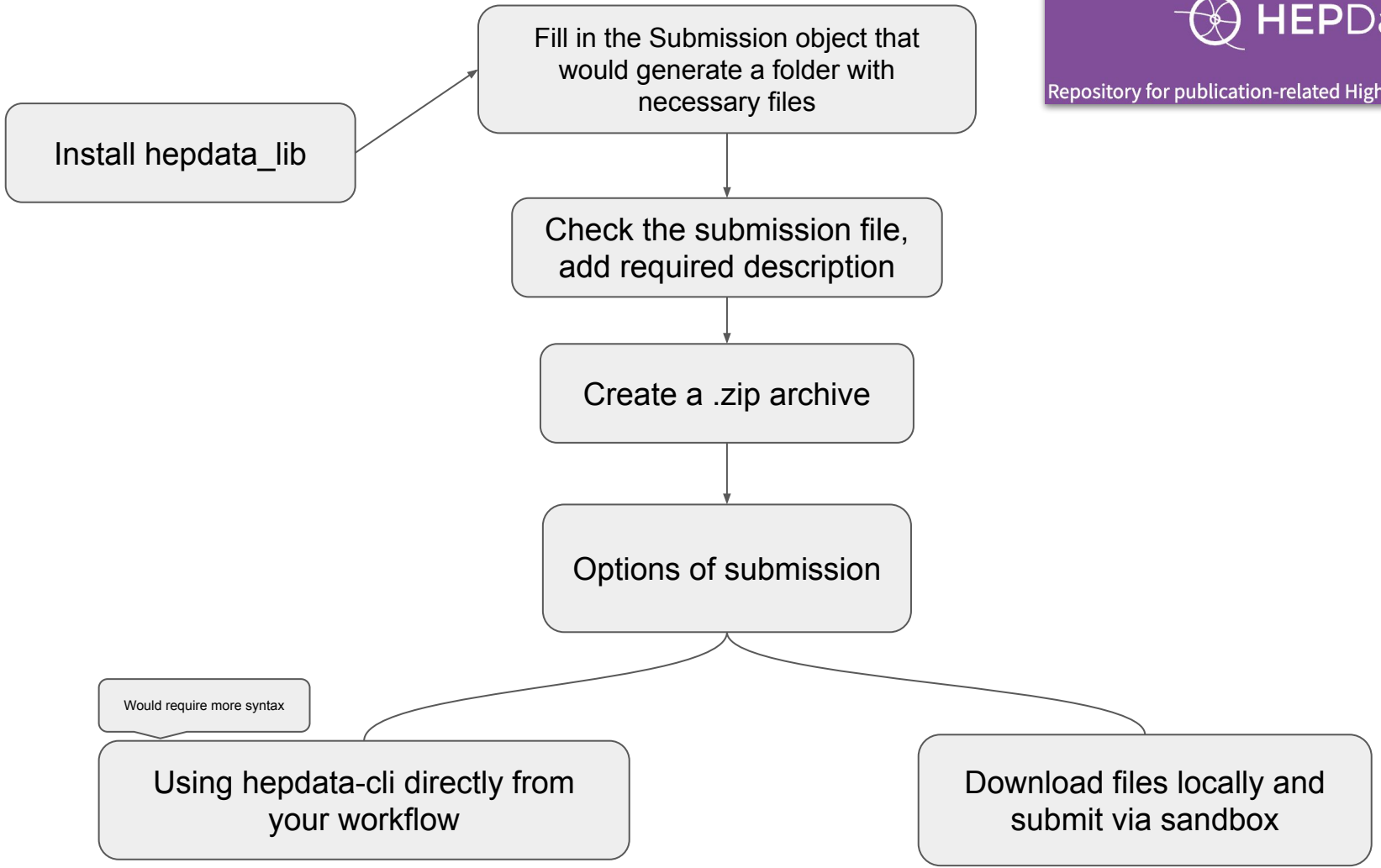
HEPDATA OUTPUT FORMATS



In addition to browsing HEPData, there are various ways to interact with it programatically and to retrieve data in different formats:

- JSON endpoints
- Data file formats
- Content negotiation

The Durham High-Energy Physics Database (HEPData) has been built up as a unique open-access repository for scattering data from experimental particle physics. HEPData is funded by a grant from the UK STFC and is based at the IPPP at Durham University.

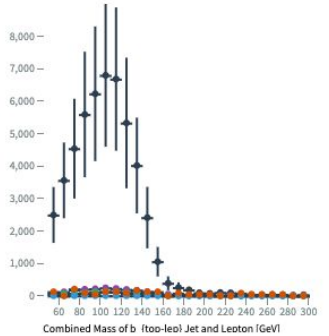

HEPData
Repository for publication-related High-Energy Physics data
Install hepdata_lib
Fill in the Submission object that would generate a folder with necessary files
Check the submission file, add required description
Create a .zip archive
Options of submission
Would require more syntax
Using hepdata-cli directly from your workflow
Download files locally and submit via sandbox

Submission of AGC histograms to HEPData

[submission example]

Combined Mass of $b_{top-lep}$ Jet and Lepton [GeV]	W+jets [Number of jets]	single top, s-channel [Number of jets]	single top, t-channel [Number of jets]	tW [Number of jets]	ttbar [Number of jets]
50.0 - 60.0	125.65 ±52.841 A symmetric error	0.96846 ±0.30465 A symmetric error	46.432 ±14.205 A symmetric error	87.655 ±28.304 A symmetric error	2488.6 ±859.4 A symmetric error
60.0 - 70.0	1.0e-06 ±58.836 A symmetric error	1.0741 ±0.35358 A symmetric error	69.311 ±21.37 A symmetric error	123.02 ±36.056 A symmetric error	3554.7 ±1165.0 A symmetric error
70.0 - 80.0	209.42 ±97.426 A symmetric error	1.5143 ±0.4862 A symmetric error	79.405 ±22.959 A symmetric error	181.36 ±54.293 A symmetric error	4532.0 ±1537.2 A symmetric error
80.0 - 90.0	83.767 ±48.293 A symmetric error	2.201 ±0.70419 A symmetric error	94.882 ±30.998 A symmetric error	202.82 ±60.124 A symmetric error	5583.2 ±1931.9 A symmetric error
90.0 - 100.0	41.883 ±46.008 A symmetric error	2.1482 ±0.62424 A symmetric error	123.82 ±38.098 A symmetric error	217.93 ±68.782 A symmetric error	6222.5 ±2072.2 A symmetric error
100.0 - 110.0	125.65 ±57.172 A symmetric error	2.43 ±0.86692 A symmetric error	127.86 ±37.682 A symmetric error	245.13 ±72.163 A symmetric error	6789.7 ±2198.5 A symmetric error
110.0 - 120.0	125.65 ±64.802 A symmetric error	2.2363 ±0.77947 A symmetric error	123.14 ±41.934 A symmetric error	227.3 ±66.311 A symmetric error	6675.1 ±2204.8 A symmetric error
120.0 - 130.0	83.767 ±50.767 A symmetric error	1.8137 ±0.71644 A symmetric error	104.3 ±32.085 A symmetric error	197.98 ±61.412 A symmetric error	5322.7 ±2010.4 A symmetric error
130.0 - 140.0	167.53 ±96.559 A symmetric error	1.4087 ±0.4944 A symmetric error	78.732 ±26.654 A symmetric error	170.47 ±52.844 A symmetric error	4012.8 ±1472.6 A symmetric error
140.0 - 150.0	125.65 ±54.784 A symmetric error	0.88042 ±0.42596 A symmetric error	45.086 ±16.784 A symmetric error	127.55 ±40.684 A symmetric error	2407.3 ±943.27 A symmetric error
150.0 - 160.0	125.65 ±56.419 A symmetric error	0.44021 ±0.13925 A symmetric error	19.515 ±8.8137 A symmetric error	79.192 ±26.8 A symmetric error	1051.2 ±450.56 A symmetric error

Visualize



Sum errors ☑ Log Scale (X) ☐ Log Scale (Y) ☐

Deselect variables or hide different error bars by clicking on them.

Variables

W+jets [Number of jets]
Summed error

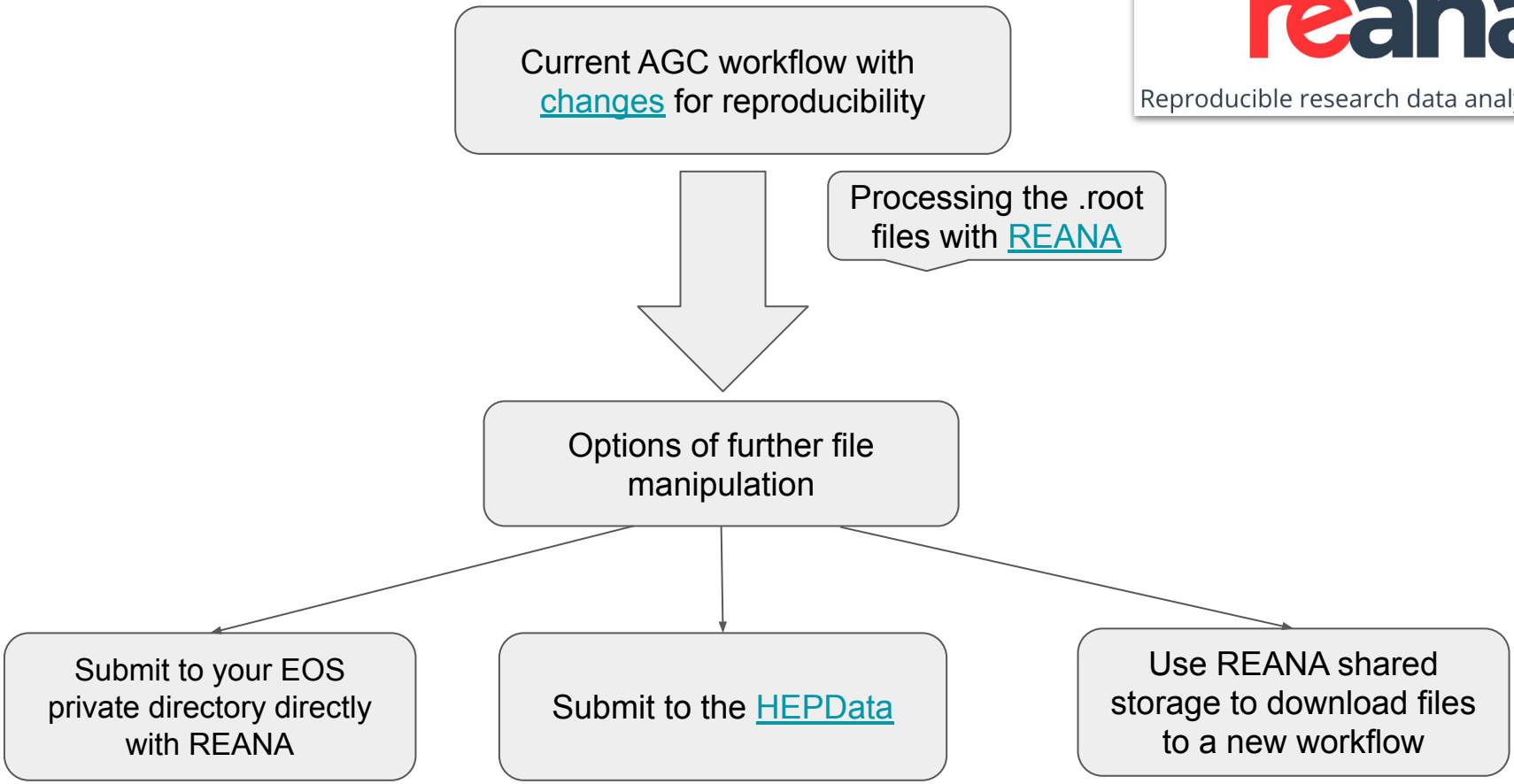
single top, s-channel [Number of jets]
Summed error

single top, t-channel [Number of jets]
Summed error

tW [Number of jets]
Summed error

ttbar [Number of jets]
Summed error

AGC and RECAST pipeline



reana
Reproducible research data analysis platform

Making a new workspace without necessity of re-running analysis

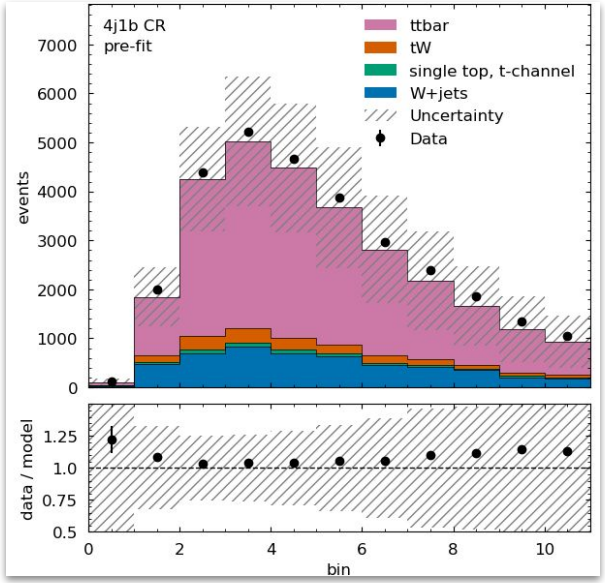
Download workspaces from HEPData/REANA

Create a patch with pyhf

New patched workspace with additional samples

Apply python-json-patch library

Apply created patch for necessary workspace.json



AGC and ServiceX

ServiceX data delivery service as an extension of AGC pipeline

- \+ Easy to modify current pipeline: we changed input files for AGC/REANA pipeline instead to use results transformed by ServiceX, which are stored in Minio object store
- \+ Using cached ServiceX files decreases time execution (few seconds per process)

Caching the queried datasets with `ServiceX`

Using the queries created with `func_adl`, we are using `ServiceX` to read the CMS Open Data files to build cached files with only the specific event information as dictated by the query.

```
if USE_SERVICEX:
    try:
        from func_adl_servicex import ServiceXSourceUpROOT
    except ImportError:
        print("cannot import func_adl_servicex, which is a required dependency when using ServiceX")
        raise

    # dummy dataset on which to generate the query
    dummy_ds = ServiceXSourceUpROOT("cernopendata://dummy", "Events", backend_name="uproot")

    # tell low-level infrastructure not to contact ServiceX yet, only to
    # return the qastle string it would have sent
    dummy_ds.return_qastle = True

    # create the query
    query = get_query(dummy_ds).value()

    # now we query the files using a wrapper around ServiceXDataset to transform all processes at once
    t0 = time.time()
    ds = utils.file_input.ServiceXDatasetGroup(fileset, backend_name="uproot", ignore_cache=utils.config["global"]["SERVICEX_IGNORE_CACHE"])
    files_per_process = ds.get_data_rootfiles_uri(query, as_signed_url=True, title="CMS ttbar")

    print(f"ServiceX data delivery took {time.time() - t0:.2f} seconds")

    # update fileset to point to ServiceX-transformed files
    for process in fileset.keys():
        fileset[process]["files"] = [f.url for f in files_per_process[process]]
```

CMS ttbar: 2% 12/787 [14:02]

ServiceX data delivery took 10xx sec. → Eventually → Data processing took few sec.

Conclusion

- Successfully implemented AGC ttbar pipeline on REANA
  - wrote multi-cascading Snakemake workflow using parameterized AGC notebook
  - run many benchmarking experiments to identify REANA performance opportunities
- Implemented submission of workflow results to HEPData
- Implement RECAST-like multi-workflow AGC pipelines in REANA
- Started to test ServiceX applicability for AGC pipelines in REANA

Further work:

- Benchmark AGC pipeline using Dask-on-REANA (new IT project under way)
- Further testing of ServiceX for AGC and ML pipelines in REANA

This work also will be presented at CHEP in October 2024

Related detailed presentations:

- AGC demo days
- Workshop on workflow languages for HEP analysis