Analysis Grand Challenge at 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

The Analysis Grand Challenge (AGC) project

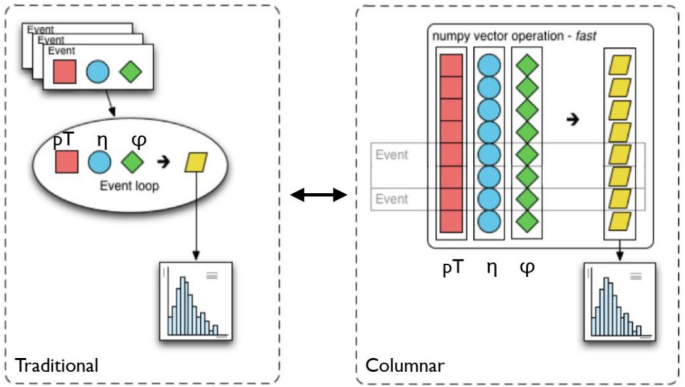
The **"Analysis Grand Challenge" (AGC)** aims **to help address the computing challenges of the HL-LHC**

- coordinated by IRIS-HEP: research and development for HL-LHC (https://iris-hep.org/)
- organized jointly with the US ATLAS & US CMS operations programs

The AGC has **two aspects:**

1. define a physics analysis task of realistic scope & scale
2. develop analysis pipelines that implements the task
   - find & address performance bottlenecks & usability concerns

Supporting new data analysis concepts for HL-LHC



Coffea Analysis Framework

ROOT Data Analysis Framework

ROOT RDataFrame

DASK

APACHE Spark™

Parsl

New columnar data analysis concepts

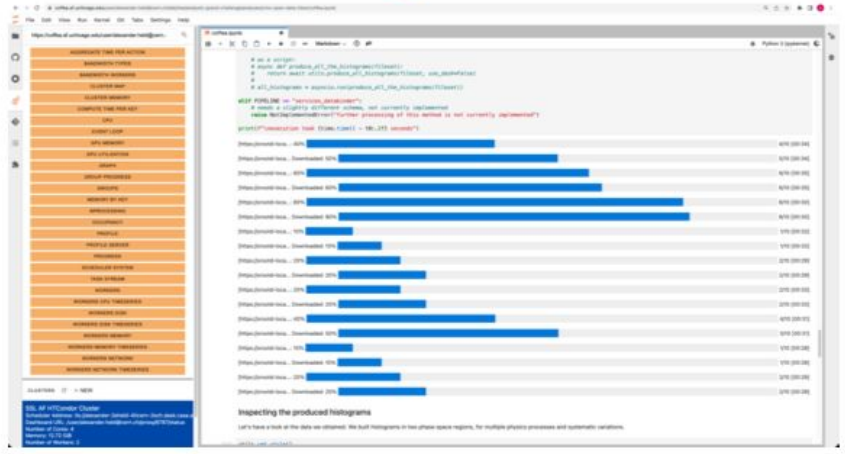
New analysis frameworks

Distributed executors

The AGC physics analysis task

Main AGC analysis task: **ttbar cross-section measurement**

- using CMS Open Data (reformatted to 2 TB of NanoAODs): anyone can participate
- key feature: different kinds of systematic uncertainties & metadata handling
- sufficient complexity to demonstrate distributed scale-out performance



Tools and services in IRIS-HEP AGC implementation

- Employing stack of **Python HEP libraries** for analysis tasks
- **ServiceX** used as data delivery service
- Execution on a **coffea-casa analysis facility and NOW @ REANA**

HEP-specific libraries used for data analysis

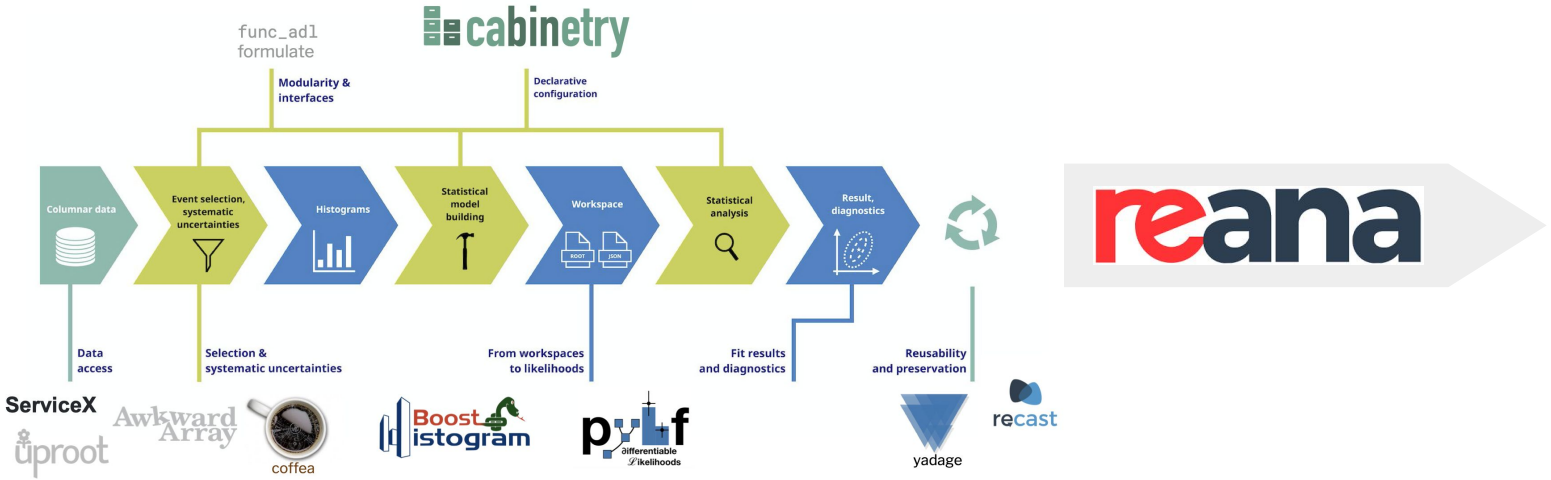
data delivery services

optional services

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

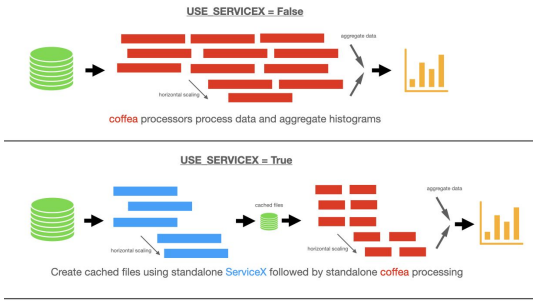
\+ **Adding analysis preservation step to AGC pipeline**



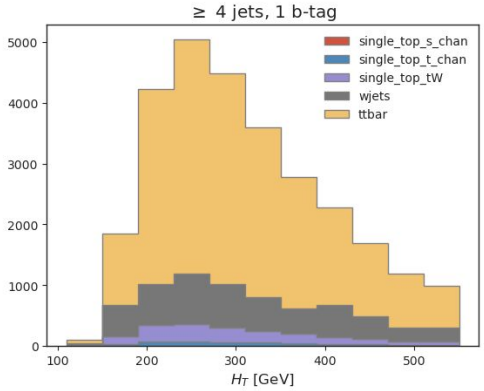
Implementation: ttbar analysis in a notebook

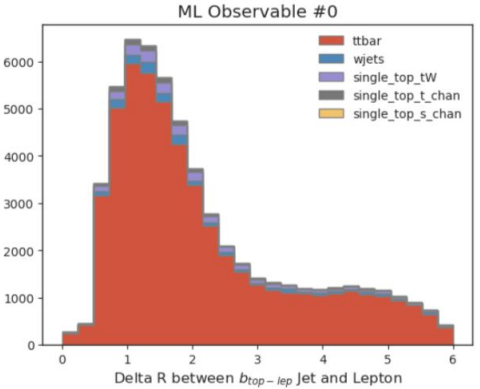
From data delivery to statistical inference in a notebook

multiple supported processing schemes

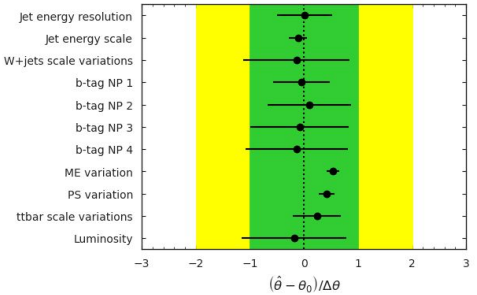


reconstructed observables

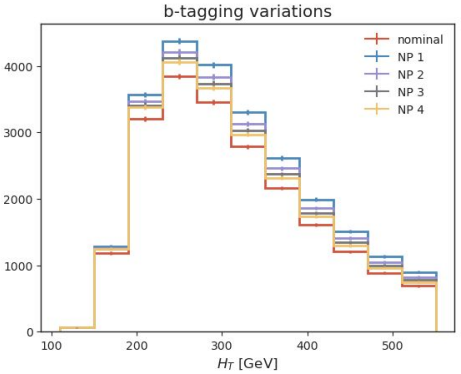




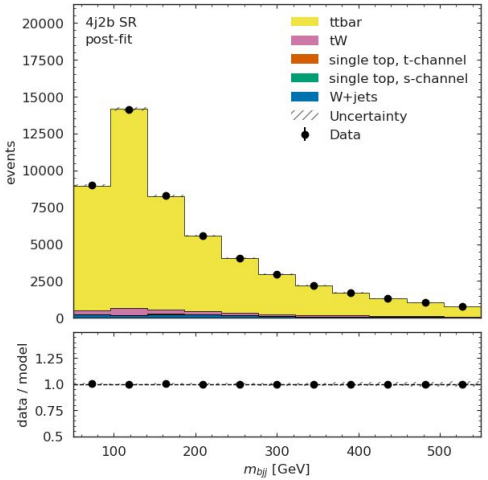
nuisance parameter pulls



systematic variations



post-fit distributions

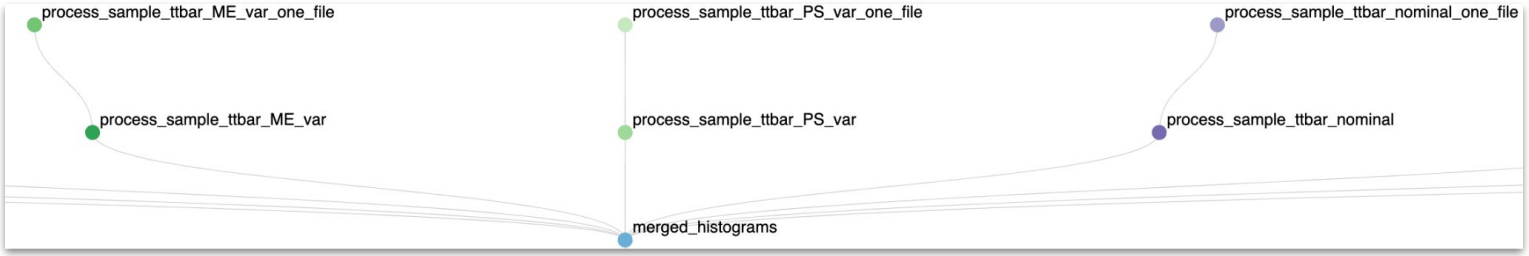


Porting Coffea analysis to Snakemake

- Our choice was to use **Snakemake workflow management system (integrated in REANA)**

- **Snakemake key feature is a “rule” description, which enables the parallelisation within REANA**, running each rule in a separate pod.
- **Snakemake allows you to create a set of rules, each one defining a “step” of your analysis.**
- In AGC case we defined each step as processing one of AGC sample (9 in total) with output file containing processed histograms for given sample
- We end up having 2 rules for one sample and final merging rule, so in total we have 19 rules which would generate 788 jobs.

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
```

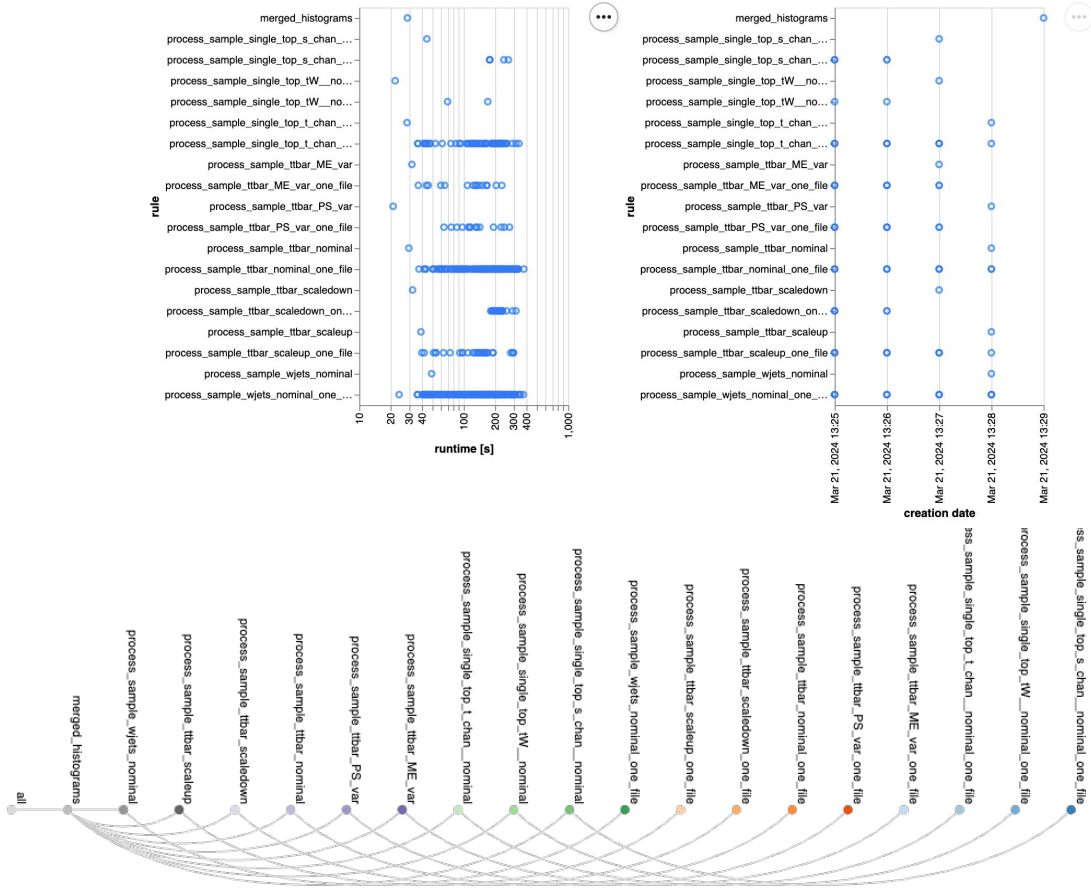
Snakemake checks the inputs and outputs in the rules to see the dependencies and order of execution

Examples of Snakemake rules

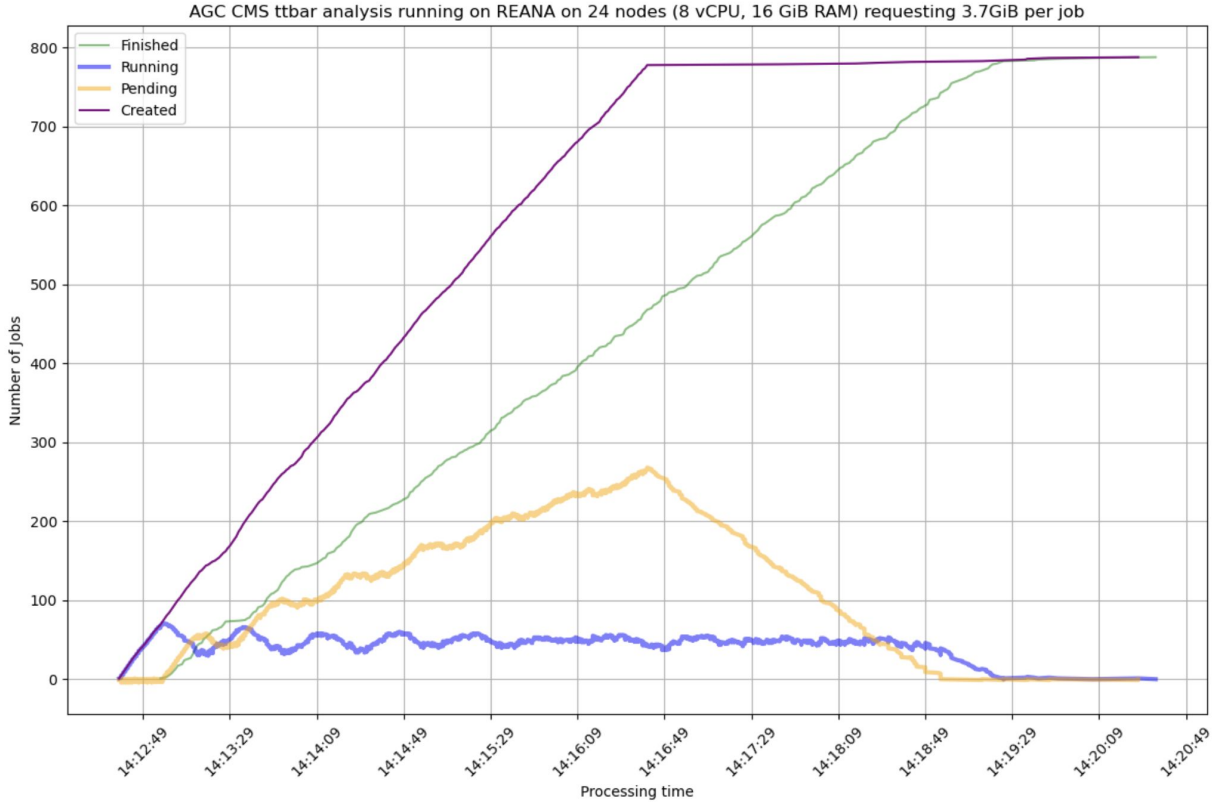
```
rule process_sample_ttbar_nominal:
    container:
        "ttbarkerberos:20240311""
    resources:
        kubernetes_memory_limit="3700Mi"
    input:
        "file_merging.ipynb",
        expand(get_file_paths("ttbar__nominal"))
    output:
        "everything_merged_ttbar__nominal.root"
    params:
        sample_name = 'ttbar__nominal'
    shell:
        "papermill file_merging.ipynb merged_nominal.ipynb -p sample_name
{params.sample_name} -k python3"
```

```
rule process_sample_ttbar_nominal_one_file:
    container:
        "ttbarkerberos:20240311"
    resources:
        kubernetes_memory_limit="3700Mi",
        kerberos = True
    output:
        "histograms/histograms_ttbar__nominal_{filename}"
    params:
        sample_name = 'ttbar__nominal'
    shell:
        "/bin/bash -l && source fix-env.sh && python prepare_workspace.py
sample_{params.sample_name}_{wildcards.filename} && papermill
ttbar_analysis_reana.ipynb sample_{params.sample_name}_{wildcards.filename}_out.ipynb
-p sample_name {params.sample_name} -p filename {url_prefix}{wildcards.filename} -k
python3"
```

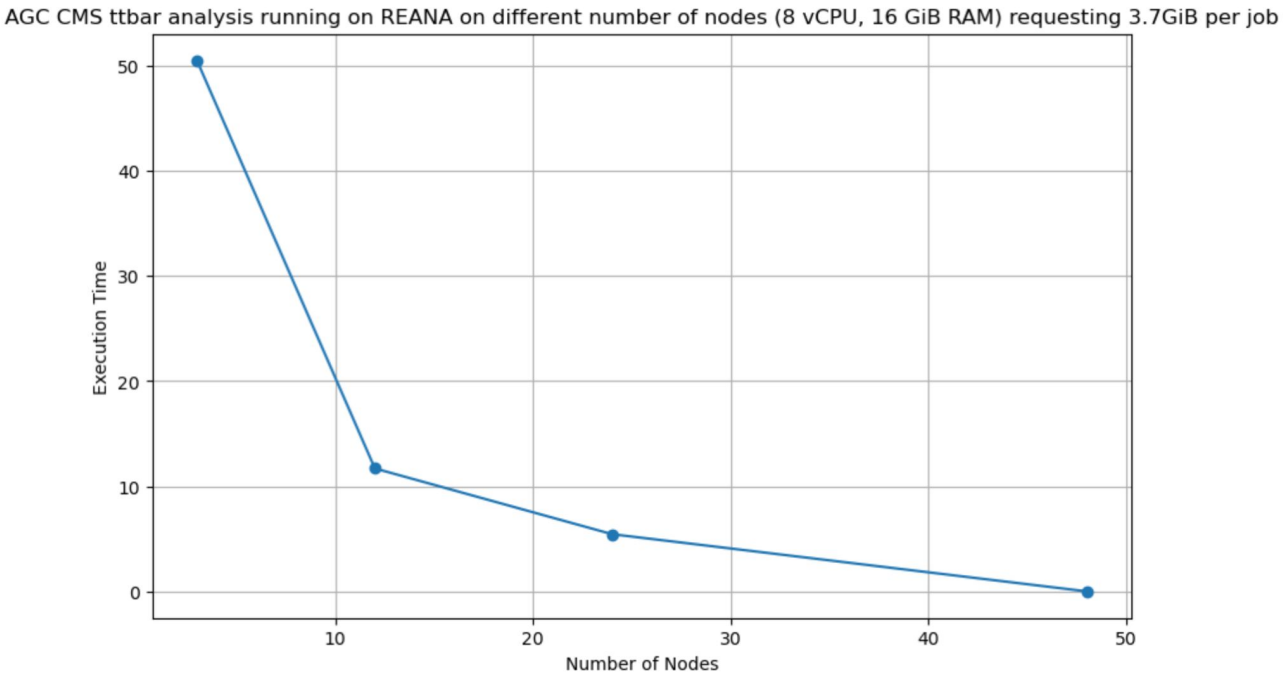
REANA AGC report



Optimising REANA k8s nodes configurations (depending on configuration)



The current execution time statistics



Issues with porting AGC IRIS-HEP implementation to be executed on REANA

- Defining efficient resources = optimisation process
- Specify the kubernetes_memory limit that exact number of jobs are running in one node.
- XCache switched off
- Dask switched off
  - Datasets at Nebraska are too far for CERN REANA instance and accessing datasets from EOS Public is sometimes not efficient
- No native Dask support in REANA (WIP!)

Conclusion

- We successfully implement the AGC ttbar analysis at REANA using Snakemake
- You need to develop the Snakemake skills to be able to make your analysis reproducible friendly.

Future Tasks

- Add the recasting step using RECAST which would allow to submit, evaluate, of additional sample which could be then merged on the final step
- Making more stress test experiments of AGC based on job memory
- Testing AGC ServiceX and Machine Learning pipelines in REANA
- Make a clear instruction for the reana-demo-agc-cms-ttbar-coffea.

Backup

AGC notebooks modification:

- Rerun the same notebook n-times but with different parameters => instead of processing all files, samples we process one sample with one file
- Firstly we parallelized each sample from fileset:

```
original_dict = fileset
selected_file = original_dict[sample_name]['files']
new_dict = {sample_name: {'files': [filename], 'metadata': original_dict[sample_name]['metadata']}}
```

- Second, parallelize each file for each sample:

```
all_histograms, metrics = run(
   fileset={sample_name: new_dict[sample_name]},
   treename=treename,
   processor_instance=TtbarAnalysis(USE_INFERENCE, USE_TRITON)
)
```

The main idea is to see the whole picture of your analysis what steps suppose to be after another and modify it on the early stages to have a separate pieces which could be count as 1 job.