Physics analysis for the HL-LHC: concepts and pipelines in practice with the Analysis Grand Challenge

Alexander Held[1,*], *Elliott* Kauffman[2], *Oksana* Shadura[3], and *Andrew* Wightman[3]

[1]University of Wisconsin–Madison, United States
[2]Priceton University, United States
[3]University of Nebraska–Lincoln, United States

Abstract. Realistic environments for prototyping, studying and improving analysis workflows are a crucial element on the way towards user-friendly physics analysis at HL-LHC scale. The IRIS-HEP Analysis Grand Challenge (AGC) provides such an environment. It defines a scalable and modular analysis task that captures relevant workflow aspects, ranging from large-scale data processing and handling of systematic uncertainties to statistical inference and analysis preservation. By being based on publicly available Open Data, the AGC provides a point of contact for the broader community. Multiple different implementations of the analysis task that make use of various pipelines and software stacks already exist.

This contribution presents an updated AGC analysis task. It features a machine learning component and expanded analysis complexity, including the handling of an extended and more realistic set of systematic uncertainties. These changes both align the AGC further with analysis needs at the HL-LHC and allow for probing an increased set of functionality.

Another focus is the showcase of a reference AGC implementation, which is heavily based on the HEP Python ecosystem and uses modern analysis facilities. The integration of various data delivery strategies is described, resulting in multiple analysis pipelines that are compared to each other.

1 Introduction

IRIS-HEP [1] is the Institute for Research and Innovation in Software for High Energy Physics. It is comprised of physicists, computer scientists, and engineers from institutes across the United States who perform research and development to address HL-LHC computing challenges by developing suitable software cyberinfrastructure. Work in the institute proceeds in various topical areas, including data reconstruction and triggering in the innovative algorithms area, the analysis systems area aiming to reduce time-to-insight and maximize physics potential, as well as the data organization, management, and access systems area.

A variety of new ideas for implementing physics analyses in the future are being pursued in IRIS-HEP and the broader community, alongside new tools and techniques which are being developed. The IRIS-HEP Analysis Grand Challenge (AGC) project [2] started in this context as an integration exercise: combine the ongoing developments to demonstrate an

*e-mail: alexander.held@cern.ch

efficient end-to-end analysis pipeline that is designed for use at the HL-LHC. There are two aspects to the AGC:

- define a physics analysis task of realistic scope and scale, representative of HL-LHC requirements,
- develop analysis pipelines that implement the task, find and address performance bottlenecks and usability concerns in the process.

1.1 Goals of the AGC project

Despite starting out as just an integration exercise, over time it became clear that the AGC project can be useful in additional ways to the broader community. For developers of software and infrastructure components, it provides a realistic testbed for evaluating user experience, interface design and performance. It is an example workload for testing modern analysis facilities and new services, allowing to explore the possibility of "interactive" analysis. Achieving an interactive turnaround time of minutes or less requires highly parallel execution with low latency, which also mandates efficient use of intermediate caching.

The AGC project has organized a series of dedicated workshops [3, 4] to showcase related work and demonstrate the envisioned analysis pipelines. These workshops provide dedicated opportunities to interact with the broader community and receive valuable input regarding the envisioned approaches showcased.

2 The AGC analysis task

The AGC focuses on the last pieces in a physics analysis pipeline, starting out from centrally produced common data samples. To capture the needs of a physicist in their practical applications, the analysis task must probe the following workflow aspects:

- extracting and filtering data from common data samples,
- calibrating objects and evaluating systematic variations,
- Machine Learning (ML) training and inference,
- statistical model building and inference,
- relevant visualizations.

A top quark pair production ($t\bar{t}$) cross-section measurement in the single lepton channel is chosen for the AGC as analysis task. It uses CMS Open Data from 2015 [5], allowing participation from anyone interested without the need for special permissions to access this data. This Open Data release is available in MiniAOD [6] format. In order to more closely correspond to the data formats expected to be used at the HL-LHC (PHYSLITE [7] for ATLAS, NanoAOD [8] for CMS), the relevant MiniAOD files are pre-converted to NanoAOD format. These converted files are available publicly in XRootD-accessible storage at the University of Nebraska–Lincoln. Details of where to find them are available in the AGC GitHub repository [9]. In total, about 1 billion events are used, corresponding to a file size of around 2 TB. The AGC analysis task, as well as implementations of it, are strictly aimed at demonstrating workflows and functionality instead of physics results.

The analysis task is set up to allow for probing the three biggest user experience pain points in physics analysis as highlighted in the Second Analysis Ecosystem Workshop report [10]: dealing with systematic uncertainties, handling metadata, and scaling a pipeline from a prototype to a sufficiently powerful facility.

Table 1. Versions of the AGC analysis task.

version	description
v0	input files in custom ntuple format
v1	input files in CMS NanoAOD format, minimal changes to analysis logic
v2 (planned)	addition of ML, expanded set of systematic uncertainties

A new addition for this conference to the AGC analysis task is a ML component. This was frequently requested by the community, owing to the ubiquitous use of ML in physics analysis. For the AGC, the ML task is the correct matching of reconstructed objects to constituents in the decay of the $t\bar{t}$ system. In practice, this implies the need to enhance the analysis pipeline: adding a ML model training step, as well as subsequent ML inference. It also changes the computational needs of the analysis pipeline: complex ML models may need dedicated specialized resources for efficient training and possibly inference.

2.1 Analysis task versions

With the AGC analysis task expanding, a versioning scheme for analysis tasks allows to keep track of what each task corresponds to. Table 1 lists the three major versions that exist. Version 0 is fully superseded. Version 1 provides a stable baseline before the addition of ML to the pipeline. Work towards finalizing version 2 is ongoing, this version includes both ML aspects and additional computational complexity via an enhanced set of systematic uncertainties. The AGC documentation webpage [11] lists additional details about the analysis task definition.

3 The AGC reference implementation

IRIS-HEP provides a Python-based reference implementation for the AGC analysis task on GitHub [9]. It uses tools and services developed within IRIS-HEP and the community, including many Scikit-HEP [12] libraries. The repository also includes information about the datasets that are being used in the AGC. Figure 1 schematically shows the pipeline and how it is implemented in practice.

The pipeline starts with an optional component: *ServiceX* [13], a data delivery service. After receiving a declarative request via *FuncADL* [14], *ServiceX* can perform filtering and return the required columns for subsequent processing. In the implementation discussed here, the output of *ServiceX* is written to object storage at an analysis facility, which serves as a cache. Subsequent executions of the analysis pipeline (without changes in the request sent to *ServiceX*) will be able to make use of this cache to speed up turnaround time. *ServiceX* can be co-located with the data it ingests for optimal performance, limiting the amount of data that needs to be transferred after the initial filtering that can already take place within *ServiceX*.

The next parts of the pipeline are steered via the *coffea* [15] framework, which controls the distributed event processing and the production of histograms. Distribution is handled in this implementation by *Dask* [16], but *coffea* supports additional alternatives. When *ServiceX* is used, *coffea* reads input files in ROOT format [17] out of the object storage via *uproot* [18]; without *ServiceX* it instead streams the remote NanoAOD files over the network. Within the code executed via *coffea*, many other libraries from the Scikit-HEP project provide functionality that is being used here. The *awkward-array* [19] library enables efficient columnar data processing, *boost-histogram* [20] and *hist* [21] are used to produce histograms from columnar

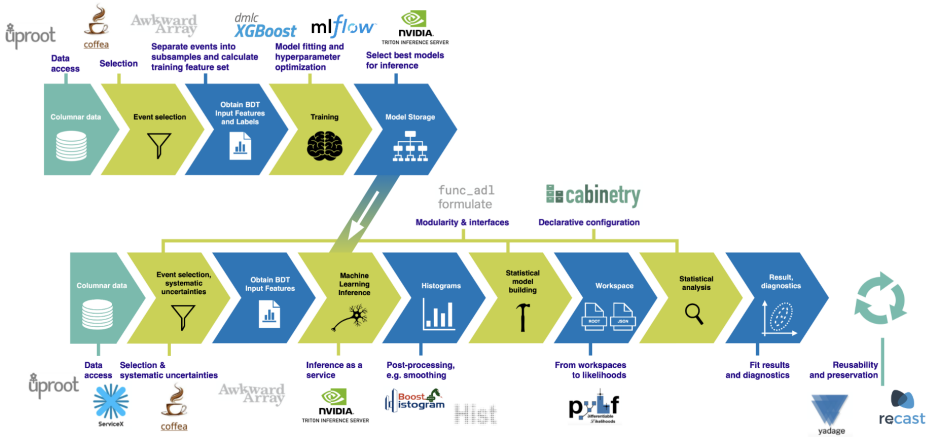

Figure 1. The reference implementation for the AGC analysis task.

data. Systematic variations are applied to data via *correctionlib* [22], which is another new addition for this conference. It provides a standardized JSON format for defining corrections and an interface for applying them. ML inference can optionally be outsourced via a *NVIDIA Triton* [23] inference server to be efficiently executed on dedicated resources. Following the production of all required histograms, *cabinetry* [24] constructs a statistical model. Statistical inference is peformed using *pyhf* [25, 26]. Additional visualization utilities are provided by *mplhep* [27].

In addition to this main processing pipeline, the top part of figure 1 shows the ML training part. It uses a similar approach to prepare the data required for training, with *coffea*, *uproot* and *awkward-array* playing major roles. A Boosted Decision Tree (BDT) is trained using *XGBoost* [28]. *MLflow* [29] provides functionality for experiment tracking to store performance metrics of different models during hyperparameter optimization. The best-performing models are uploaded for use via *NVIDIA Triton*.

3.1 Reference implementation versions

Several versions of the AGC reference implementation exist. In the versioning scheme used, the major version corresponds to the version of the analysis task as shown in table 1. The first available version, v0.1, is used for the benchmarking results presented at the ACAT 2022 conference [30]. Following that, v0.2 improves the data delivery pipeline: instead of downloading the output of *ServiceX*, in this version *coffea* streams it directly out of the object storage to which *ServiceX* writes its output.

Version v1.0 implements analysis task v1, handling the switch to input files in CMS NanoAOD format, with otherwise minimal changes to the analysis logic. Subsequent releases in the v1 series fix minor bugs in the implementation and expand functionality for benchmarking purposes.

A version 2.0 implementation is planned to be included once the corresponding analysis task definition has been finalized. The description of the implementation in these proceedings corresponds to a task version that already includes aspects of the upcoming version 2.0, in particular the ML parts of the pipeline.

4 Connections to analysis facilities and benchmarking

The majority of AGC reference implementation development and testing happens on analysis facilities deployed in the *coffea-casa* [31] model. They provide the required software environment and services which are used in the AGC reference implementation (such as *ServiceX* and *NVIDIA Triton*). Users of *coffea-casa* facilities are presented with a JupyterLab [32] interface and can seamlessly scale to computing resource provide via a batch system by using *Dask*.

Work on the AGC project includes performance evaluations of AGC implementations in order to identify and address bottlenecks. First results are documented in proceedings from the ACAT 2022 conference [30]. They demonstrate efficient scaling to hundreds of CPU cores for distributed execution and an efficient resource usage by scheduling with *Dask*.

The AGC reference implementation includes optional handles that allow altering the computational cost of the analysis task, as well as the amount of data that needs to be read from input files. These allow probing additional types of workloads for benchmarking purposes, simulating analyses that are more likely to be I/O- or CPU-bound.

5 Future plans

The next major goal for the AGC project is finalizing version v2 of the AGC analysis task and providing a corresponding reference implementation. It will feature a ML component, which has already been implemented, as well as an extended set of systematic uncertainties to handle and correspondingly a larger amount of data to ingest and process.

A longer term goals is the implementation of a differentiable analysis pipeline to investigate the benefits of gradient-based automatic analysis optimization. Another target for the AGC is the demonstration of a "column joining" mechanism. This refers to dynamically enhancing datasets used by physicists for analysis with missing information only available in parent datasets. An example of this is automatically joining together event information provided by NanoAOD input files with additional information only found in MiniAOD datasets and presenting them to the user in a unified way.

6 Conclusions

The AGC project develops and studies workflows for physics analysis at the HL-LHC. It is centered around a physics analysis task and implementations thereof, which provide a central gathering point for the broader community and context for discussions about future analysis workflows. The analysis task is based on using CMS Open Data to perform a $t\bar{t}$ cross-section measurement, which has been extended to include a frequently requested ML component. IRIS-HEP provides a Python-based reference implementation for this analysis task. All required data and the reference implementation are accessible via a GitHub repository.

This work was supported by the U.S. National Science Foundation (NSF) Cooperative Agreement OAC-1836650 (IRIS-HEP).

The AGC is made possible thanks to the help of a large number of people working on many different projects. Thank you in particular to the teams behind: coffea-casa, Scikit-HEP, coffea, IRIS-HEP Analysis Systems, ServiceX, IRIS-HEP DOMA, IRIS-HEP SSL, and the CMS Data Preservation and Open Access (DPOA) group.

References

[1] P. Elmer, M. Neubauer, M.D. Sokoloff, *Strategic Plan for a Scientific Software Innovation Institute (S2I2) for High Energy Physics* (2017), arXiv:1712.06592 [physics.comp-ph]

[2] A. Held, O. Shadura, PoS **ICHEP2022**, 235 (2022)

[3] A. Held, O. Shadura (organizers), *IRIS-HEP AGC Tools 2021 Workshop*, https://indico.cern.ch/event/1076231/ (2021)

[4] A. Held, O. Shadura (organizers), *IRIS-HEP AGC Tools 2022 Workshop*, https://indico.cern.ch/event/1126109/ (2022)

[5] CMS Data preservation and open access group, *Getting Started with CMS 2015 Open Data*, https://opendata.cern.ch/docs/cms-getting-started-2015 (2022)

[6] G. Petrucciani, A. Rizzi, C. Vuosalo, on behalf of the CMS Collaboration, Journal of Physics: Conference Series **664**, 072052 (2015)

[7] J. Elmsheuser et al., EPJ Web Conf. **245**, 06014 (2020)

[8] A. Rizzi, G. Petrucciani, M. Peruzzi (CMS), EPJ Web Conf. **214**, 06021 (2019)

[9] A. Held, O. Shadura, M. Feickert, J. Chakraborty, M. Proffitt, K. Choi, A. Novak, D. Koch, M. Adamec, S. Chopra et al., *Analysis Grand Challenge*, https://doi.org/10.5281/zenodo.7274936

[10] G.A. Stewart, P. Elmer, G. Eulisse, L. Gouskos, S. Hageboeck, A.R. Hall, L. Heinrich, A. Held, M. Jouvin, T.J. Khoo et al., *HSF IRIS-HEP Second Analysis Ecosystem Workshop Report* (2022), https://doi.org/10.5281/zenodo.7003963

[11] E. Kauffman, A. Held, O. Shadura, *Analysis Grand Challenge Documentation*, https://agc.readthedocs.io/en/latest/ (2023)

[12] E. Rodrigues et al., EPJ Web Conf. **245**, 06028 (2020)

[13] B. Galewsky, R. Gardner, L. Gray, M. Neubauer, J. Pivarski, M. Proffitt, I. Vukotic, G. Watts, M. Weinberg, EPJ Web Conf. **245**, 04043 (2020)

[14] M. Proffitt, G. Watts, EPJ Web Conf. **251**, 03068 (2021)

[15] L. Gray, N. Smith, B. Tovar, A. Novak, J. Chakraborty, P. Fackeldey, N. Hartmann, G. Watts, D. Thain, G. Stark et al., *coffea*, https://doi.org/10.5281/zenodo.3266454

[16] Dask Development Team, *Dask: Library for dynamic task scheduling*, https://dask.org (2016)

[17] R. Brun, F. Rademakers, Nucl. Instrum. Meth. A **389**, 81 (1997)

[18] J. Pivarski, H. Schreiner, A. Hollands, P. Das, K. Kothari, A. Roy, J. Ling, N. Smith, C. Burr, G. Stark, *Uproot*, https://doi.org/10.5281/zenodo.4340632

[19] J. Pivarski, I. Osborne, I. Ifrim, H. Schreiner, A. Hollands, A. Biswas, P. Das, S. Roy Choudhury, N. Smith, M. Goyal, *Awkward Array*, https://doi.org/10.5281/zenodo.4341376

[20] H. Schreiner, H. Dembinski, A. Goel, J. Gohil, S. Liu, J. Eschle, C. Maji, A. Novak, C. Burr, D. Davis et al., *boost-histogram*, https://doi.org/10.5281/zenodo.3492034

[21] H. Schreiner, S. Liu, A. Goel, *hist*, https://doi.org/10.5281/zenodo.4057112

[22] N. Smith, *Correctionlib* (2022), https://doi.org/10.5281/zenodo.7129907

[23] NVIDIA Corporation, *Triton Inference Server: An Optimized Cloud and Edge Inferencing Solution.*, https://github.com/triton-inference-server/server

[24] A. Held, M. Feickert, H. Schreiner, L. Henkelmann, A. Hollands, E. Kauffman, N. Simpson, R. Mueller, *cabinetry*, https://doi.org/10.5281/zenodo.4742752

[25] L. Heinrich, M. Feickert, G. Stark, *pyhf*, https://doi.org/10.5281/zenodo.1169739

[26] L. Heinrich, M. Feickert, G. Stark, K. Cranmer, Journal of Open Source Software **6**, 2823 (2021)

[27] A. Novak et al., *mplhep* (2022), https://doi.org/10.5281/zenodo.3766157

[28] T. Chen, C. Guestrin, *XGBoost: A Scalable Tree Boosting System*, in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), https://doi.org/10.1145/2939672.2939785

[29] M.A. Zaharia, A. Chen, A. Davidson, A. Ghodsi, S.A. Hong, A. Konwinski, S. Murching, T. Nykodym, P. Ogilvie, M. Parkhe et al., IEEE Data Eng. Bull. **41**, 39 (2018)

[30] O. Shadura, A. Held, *First performance measurements with the Analysis Grand Challenge*, in *21th International Workshop on Advanced Computing and Analysis Techniques in Physics Research: AI meets Reality* (2023), https://doi.org/10.48550/arXiv.2304.05214

[31] M. Adamec, G. Attebury, K. Bloom, B. Bockelman, C. Lundstedt, O. Shadura, J. Thiltges, EPJ Web Conf. **251**, 02061 (2021)

[32] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay et al., *Jupyter Notebooks - a publishing format for reproducible computational workflows*, in *Positioning and Power in Academic Publishing: Players, Agents and Agendas* (2016), https://dx.doi.org/10.3233/978-1-61499-649-1-87