Analysis Grand Challenge

Alexander Held (University of Wisconsin–Madison)
Oksana Shadura (University Nebraska–Lincoln)

Nov 15, 2022

IRIS-HEP / Ops Program Analysis Grand Challenge Planning
https://indico.cern.ch/event/1216989/

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

*How to deploy the software stack*
An AGC implementation: software stack
Involves large number of packages from IRIS-HEP and partners

Analysis specific frameworks and packages (available in Docker container)

Data delivery service (k8s)

Optional services (k8s)
Analysis pipeline

- **Pipeline setup**
  - ServiceX delivers columns following declarative `func_adl` request
  - coffea orchestrates distributed event processing & histogram production
    - Using uproot, awkward-array, hist
  - Visualization with hist & mplhep
  - Statistical model construction with cabinetry & inference with pyhf
- **Everything is openly developed** ([IRIS-HEP AGC repository](#))
  - Including categorization of datasets in terms of role in AGC demonstrator
- Will be executed on various partner facilities: University Nebraska-Lincoln, UChicago, FNAL, BNL, others

---

**Other (partial) AGC implementations:**

- **ROOT RDF** (Andrii Falko, Enrico Guiraud): [andriiknu/RDF/](#)
- **Julia** (Jerry Ling): [Moelf/LHC_AGC.jl](#)
Analysis libraries

Deployment requirements

- Many Python libraries from Scikit-HEP + surrounding ecosystem
  - `uproot`, `awkward`, `pyhf`, `cabinetry`, ...
    - Easily pip / conda installable
    - You can use **Docker image for testing on your facility**: `hub.opensciencegrid.org/iris-hep/analysis-systems-base`
coffea requirements

Deployment requirements

- **Easily pip / conda installable**
  - No extra (non-Python) dependencies

- **Scaling via Dask / WorkQueue / Parsl / Spark / Python futures**
  - Dask allows you easily to connect to almost any **batch system**
    - You probably will need to have your custom module allowing to hide configuration from user and integrate in the existing setup (see *lpcjobqueue*, *coffea_casa*, *dask_lxplus*, and many more).
    - You can also easily scale over Kubernetes resources
  - Resource requirements:
    - Worker configuration at least 2GB/core
Data management tools

Relying on DOMA R&D for fast physics analysis turnaround

Expect **key contributions to improved performance** from three IRIS-HEP DOMA projects:

- **XCache** — XRootd file-based caching proxy used for regional / site caches to store requested on-demand datasets (reducing latency & WAN traffic)
- **ServiceX** — data extraction and delivery service (“column-on-demand” service)
- **Skyhook DM** — an extension of the Ceph distributed storage for the scalable storage of tables and for offloading common data management operations to them (selection, projection, aggregation, and indexing, as well as user-defined functions)
ServiceX: Column-on-demand service

- **ServiceX** could bring further performance improvements
  - Initial event filter reduces number of events that need to be processed again
  - Repeated columnar processing can read cached data

- New interesting workflow to be investigated: column addition from parent MiniAOD/PHYS datasets

Columns from NanoAOD / PHYSLITE and request column from MiniAOD / PHYS
ServiceX requirements

Deployment requirements

- Kubernetes / OKD
  - A decent number of cores is needed for the ServiceX to scale up transformers.
- While not strictly required, **XCache** is essential for a performant data access
- An object store to send results to
  - Anything that supports S3 API will work
  - Missing that, a performant PersistentVolume of at least 1TB will be needed for a built-in MinIO
XCache setup

- At least one node
- Preferably ~10-20TB of **NVMe storage**
- **Good NIC** (optimally 50Gbps)
- Configured for block level caching
- Capable of accessing all of the collaboration’s data

**ATLAS approach** *(details) @ UChicago*
- SLATE managed XCache
- 17x1.1TB SSD, 2x25 Gbps
- Registered in Rucio
- ServiceX ask Rucio for paths to input data
- Rucio returns distance ordered and XCache prepended paths to all the active file replicas
- ServiceX will (re)try three times each replicas in order of distance

**CMS approach @ UNL**
- Dedicated node (using cms-xcache container OSG)
- 88TB RAID60 array (two 44TB RAID6 arrays combined in RAID0) and a 10Gb NIC *(moving to new hardware!)*
- ServiceX asks Rucio for input files gLFNs, and changes server with XCache server
- XCache is a part of CMS XCache redirection tree that connects to AAA redirection tree
- Retries are handled by AAA
IRIS-HEP Demo day
IRIS-HEP Demo Day
16 Dec 2022, 17:00 CET / 10:00 Central

- Inspired by *Demo Day* idea @ Nvidia, Dask
  - Demonstrate progress on key items
- We hope to run IRIS-HEP demo day every two months
- Open meeting, you are *welcome to join!*
ACAT 2022:

AGC performance results
AGC + local files
Behavior in idealistic setup

UChicago Coffea-casa AF @ UChicago ATLAS AF (coffeawithFuturesExecutor): reading locally stored files and scaling on local machine (hyperthreading after 48 cores)
AGC: scale-out

Processing 1B events

scaling with number of events processed

UNL Coffea-casa AF @ UNL CMS Tier-2 (Coffea with DaskExecutor): stable scaling up to 1B events on Tier-2 job queue and efficient scheduling
Results: scaling behavior

I/O and number of cores

UNL Coffea-casa AF @ UNL CMS Tier-2 (Coffea with DaskExecutor): stable scaling to 400 cores events with increasing number of branches (bigger fraction of data to read)
Goals for IRIS-HEP year 5 & longer term ideas
Year 5: minimal / stretch / ideal scenario

Minimal / ideal AGC setup for year 5

- Minimal scenario
  - AGC showcase event in spring 2023 together with AF partners
  - Extensive documentation for analysis task to allow re-implementations

- Stretch goals
  - Performance tuning (DOMA/AS)
  - Extended input size & analysis task (more systematics, more histograms, …)
  - AGC in REANA / related work on re-use and re-interpretations (AS)
  - ML component (already in progress)
    - Including exploring GPU integration at AFs into pipeline

- Longer term ideas
  - Processing implementation improvements (systematics handling, use of correctionlib)
  - Improved ServiceX + coffea integration
  - Analysis extension: search for BSM signal in same phase space
  - ATLAS / CMS-specific versions
Towards AGC execution workshop

- Our idea is to have an **AGC Execution Workshop** some time after CHEP 2023 (end of May?)
- Inviting everyone who is interested to share their setup and to present the results
  - Interesting combinations of hardware, network site configurations
  - Any type of “combinatorics” of AGC analysis implementation / components setup
  - The chance to publicize your site to physics analysis community :-)

- Stay in touch: [analysis-grand-challenge@iris-hep.org](mailto:analysis-grand-challenge@iris-hep.org) (sign up: [google group link](https://groups.google.com)), and please also feel free to contact us if you’d like to get involved or have any questions!
Backup
AGC: give it a try!
We are making it easy for you to try out our setup

- **One click** to get PyHEP notebook in Binder environment
  - **Try it out today!**
- You can also use the [UNL Open Data coffea-casa](https://example.com)
  - Or **SSL** (ATLAS members), or your favorite facility
  - This allows you to scale up (limited on Binder)
  - Everything is available in the [AGC repository](https://example.com)
The IRIS-HEP Analysis Grand Challenge (AGC) has two components:

- Defining a physics analysis task of realistic HL-LHC scope & scale

- Developing an analysis pipeline that implements this task
  - Finding & addressing performance bottlenecks & usability concerns

You can (for example) take an analysis task and develop a different implementation, take a pipeline and try it with a new analysis task, or adopt task & implementation and run it on your favorite facility.
AGC: how we envisioned it initially
An “integration exercise” for IRIS-HEP

- Demonstrate method for handling HL-LHC data pipeline requirements
  - Large data volumes + bookkeeping
  - Handling of different types of systematic uncertainties
  - Use of reduced data formats (PHYSLITE / NanoAOD), aligned with LHC experiments

- Aiming for “interactive analysis”: turnaround time of ~minutes or less
  - Made possible by highly parallel execution in short bursts, low latency & heavy use of caching

- Specify all analysis details to allow for re-implementations and re-use for benchmarking

- Execution on Analysis Facilities
AGC: analysis task
Community benchmark

- Analysis task: **ttbar cross-section measurement** in single lepton channel
  - Includes simple top reconstruction
  - Captures relevant workflow aspects and can easily be extended
    - E.g. conversion into a BSM search
  - Analysis task prominently features handling of systematic uncertainties

- Analysis is based on **Run-2 CMS Open Data** (~400 TB of MiniAOD available)
  - Open Data is crucial: everyone can participate
  - Currently using 4 TB of ntupke inputs (pre-converted, ~1B events before cuts)

- Goal of setup is showing **functionality**, not discovering new physics
  - Want to capture workflow; use made-up tools for calibrations & systematic uncertainties
AGC: what we mean by “analysis”

Typical steps in an analysis workflow

- Start from centrally produced common data samples
- Perform all subsequent steps (in a reproducible way)
  - Extract relevant data
  - (Re-) calibrate objects & calculate systematic variations
  - Filter events & calculate observables
  - Histogramming (for binned analyses)
  - Construct statistical model & perform statistical inference
  - Visualize results & provide all relevant information to study analysis details
Adding ServiceX to the mix

Benefits of caching

- Investigating different data pipelines
- Data delivered by ServiceX can be filtered and is cached locally
  - Subsequent runs can hit (filtered) cache for significant speedup
## What currently runs where?

*(please help us update the gaps)*

<table>
<thead>
<tr>
<th>Model</th>
<th>BNL</th>
<th>FNAL</th>
<th>SLAC</th>
<th>UNL</th>
<th>UChicago</th>
</tr>
</thead>
</table>
| **basic coffea** *(e.g. IterativeExecutor)*  
| *-> notebook* with USE_DASK = False | ✓   | ✓    | ✓    | ✓   | ✓        |
| **coffea** scaling *(e.g. with Dask)*  
| *-> notebook* with default settings* | ✓   | ✓    | ✓    | ✓   | ✓        |
| **standalone ServiceX**  
| *-> notebook* (no configuration) | ✓   | ✓    | ✓    | ✓   | ✓        |
| **ServiceX+coffea+scaling**  
| *-> notebook* with PIPELINE = "servicex_processor" | ✓   | ✓    | ✓    | ✓   | ✓        |
| **XCache support**  
| | ✓ | ✓ | ✓ (some performance caveats, to be understood) | ✓ | ✓ |
AGC implementations

Community effort

- **coffea**: iris-hep/analysis-grand-challenge/
- **ROOT RDF** (Andrii Falko, Enrico Guiraud): andriiknu/RDF/
- **Julia** (Jerry Ling): Moelf/LHC_AGC.jl