U.S. CMS Managed Analysis Facilities

HSF WLCG Virtual Workshop 2020

Mat Adamec, Ken Bloom, Oksana Shadura, University of Nebraska, Lincoln

Garhan Attebury, Carl Lundstedt, Derek Weitzel, University of Nebraska Holland Computing Center

Mátyás Selmeci, University of Wisconsin, Madison

Brian Bockelman, Morgridge Institute
Analysis facilities: how we understand it

- **People, software / services, and hardware**
- **Services** includes:
  - Access to experimental data products
  - Storage space for per-group or per-user data (often ntuples)
  - Access to significant computing resources
- **Physics software**: ROOT and the growing Python-based ecosystem
- **Computing hardware**: available/new CPUs and disks (maybe GPUs)
  - The main goal is to reuse existing resources

B. Bockelman, Analysis Facilities for the HL-LHC, Snowmass 2020

Trending towards column-wise (tidy/big data) analysis:

One of examples: https://github.com/CoffeaTeam/coffea
Analysis facilities: prototypes

- **Two AF facilities** with the possible outcome of adding more sites as soon as we gain experience

CMSAF @T2 Nebraska
“Coffea-Casa”
[https://cmsaf-jh.unl.edu](https://cmsaf-jh.unl.edu)

Elastic AF @ Fermilab

- **Q4 2020** - Invite first users to test “alpha” version of UNL AF (“coffea-casa”)
- **Q4 2020** - Make “coffea-casa” products (Helm charts, modules) deployable in any other AF facility
  - Expected first test deployment of FNAL Elastic AF during 2021
- **Q4 2020** - Finalize testing of ServiceX@UNL AF
- **Q1 2021** - Deploy and test data delivery with Skyhook at UNL AF
Analysis Facility @ T2 Nebraska

Per-user 8 Core “CMS Analysis pod” created on login (Dask scheduler container and Dask worker sidecar container)

Can scale up to available HTCondor slots on the T2 resource
Analysis facilities: why we use Dask?

- Dask provides flexible library for parallel computing in Python
- Builds on top of the python ecosystem (e.g. Numpy, Pandas, Scikit-Learn and etc.)
- Dask exposes lower-level APIs letting to build custom systems for in-house applications (!)
- Integrates with HPC clusters, running a variety of schedulers including SLURM, LSF, SGE and HTCondor via “dask-jobqueue”
- This allows us to create a user-level interactive system via queueing up in the batch system

Dask can be used inside Jupyter or you can simply launch it through Jupyter and connect directly from your laptop
Enabled token authentication in HTCondor infrastructure

We are using a highly customized “CMS Analysis” container with all the necessary dependencies

Pod customization hook to create secrets and services - pod can expose the Dask scheduler to the outside world and can authenticate with services like HTCondor and XRootD

Developed a custom XRootD client plugin enabling whenever the prefix root://xcache/ is used, hostname is replaced with the correct one for the local site (using environment variables) and token authorization is automatically used & embedded in the URL

Security - TLS enabled communication between workers and scheduler

CoffeaCasaCluster: HTCondorCluster integration for Dask to allow auto-scaling out to the local HTCondor pool

All of this is being incorporated into a Helm chart - many rough edges, but it will be portable to other sites

Integration of XRootD - each pod's unique secret includes and auto-generated macaroon authorizing the pod to access files at the site XCache server
Analysis facilities: “ideal” workflow

Request data “over night” via data delivery service (filter, add specific columns, optimise data layout for columnar analysis in AF)

Request addition to existing NanoAOD (request info from MiniAOD /correct existing branches)

Analyse custom NanoAOD in AF (as well in a batch from AF)

Get results on your laptop from AF

We expect a higher data rate I/O (at least order of magnitude) and latency will be more important in this case!
Analysis facilities: data lake data transfer

AFs are one of the components of Datalake’s distribution model

- For AF access to data from processing elements inside the lake is mediated either via caches (implemented via XCache) or streaming directly from one of the data lake origins

B. Bockelman, IRIS-HEP WLCG DOMA F2F

CMSAF @T2 Nebraska “Coffea-casa”
https://cmsaf-jh.unl.edu
Data delivery services: Service X

- **ServiceX provides user level ntuple production**
- Converts experiment-specific datasets to columns
- Extracts data from flat ROOT files
- Enable simple cuts or simple derived columns, as well as specified fields
  - Heavy-weight analysis will still happen via some separate processing toolchain (like CRAB)
- ServiceX already supports NanoAODs, and will also support MiniAOD extension to end-user “ntuples” derived from NanoAOD

IRIS-HEP Scalable System Laboratory [https://iris-hep.org/ssl](https://iris-hep.org/ssl)
Data delivery services: ServiceX + Skyhook

- The Skyhook DM project has shown the ability to ingest ROOT files (particularly, CMS NanoAOD) and convert event data to the internal object-store format
- Ceph-side C++ plugins transition from on-disk format to desired memory format
- Uses Dask workers to distribute data to clients
- Data delivered as Arrow tables and (optionally) presented as Arrow Dataframes

B. Bockelman, IRIS-HEP WLCG DOMA F2F

Next to be deployed @coffea-casa

CROSS, UCSC [https://github.com/uccross](https://github.com/uccross)
Coffea Team (FNAL) has an idea to design a scale-up mechanism for coffea users that removes the need to curate skims and re-run expensive algorithms over and over.

- Shared input cache at column granularity
- Derived columns declared, only constructed and cached on access
- Unified metadata and dataset schema database

Components (as it stands):

- Columnservice REST API
  - Fully async python backend
  - Using starlette & fastapi
  - Uses dask for ROOT file indexing work
- Dataset/columnset index
  - MongoDB
- Object store
  - Minio or other S3
  - Shared filesystem (yuck)
- ColumnClient
  - Python singleton client to:
    - REST API
    - xrootd federation (“FileCatalog”)
Storage expectations for AF @ T2

- AF should provide a columnar data store
- Expected to provide a fast data ingest over WAN at up to 400Gbps
- AF could have a mix of HDD and NVMe at the relative size of 80/20 or 70/30, providing really fast random access in NVMe

Frank Wuerthwein, UCSD/SDSC “T2 UCSD Storage Expectations for HL-LHC”
Analysis Facility @ T2 Nebraska: next steps

We are looking for the volunteers (other sites) to try our developments!
Conclusions

- **R&D:** there is exist a growing analysis ecosystem for HL-LHC, which we try to test and deploy in dedicated analysis facilities
  - *Our main goal is to reuse existing resources*

- **I/O:** For HL-LHC era, we expect a higher data rate I/O (at least order of magnitude) and latency will be more important in this case
  - *Low latency data access:* AF are expected to require low latency random access media to achieve best performance (via data access patterns)
    - Required as a part of uncertainties for WAN IO needs at Tier-2 centers hosting AF
    - A trade off investments in disk space in caching infrastructure against network bandwidth use

- **Reusability:** AF expected to support extraction of user defined data formats to migrate onto laptops, desktops, workstations at home institutions or at home
Thank you for your attention!

Many thanks to the other teams (IRIS-HEP SSL, IRIS-HEP DOMA and Skyhook, Coffea Team) for materials
Backup slides
Analysis facilities: data analysis challenges

- Typical CMS analyses of MiniAOD in Run 2 reach rates $O(10)\text{Hz}$
- Columnar analysis of NanoAOD $\sim O(1\text{kHz})$ (both per hyperthread)

Analysis bottleneck in Run2 MiniAOD analyses is the creation of fast user ntuples

Expect 50% of analyses to use NanoAOD, most of the rest will use MiniAOD (from resource planning for HL-LHC)

The AF should help to assist with 90% of analyses using NanoAOD by merging parts or derived from MiniAOD into Nano (automatically, without the intervention of the end-user)
Analysis facilities: expected scaling

**Today**
- CRAB @ $O(10Hz)$ per thread on 1000 threads
  - 10kHz to analyze
  - 10 Billion events (~50% of annual) takes 1 M seconds
  - ~2 weeks

**HL-LHC**
- AF @ $O(1kHz)$ per thread on 1000 threads
  - 1MHz to analyze
  - 50 Billion events (~50% of annual) takes 50,000 seconds
  - ~1 day

*With the same resources, a much larger sample size can be supported for interactive analysis due to the inherent speed-ups in the technologies chosen*
Analysis facilities: expected scaling

If we use a sample that is small enough to allow for interactive analysis in NANO (~ 50 Million events), it should be useable as driver for data delivery service to add objects from MINI overnight!

If we can transition to use of NANO as primary driver then data volumes should be manageable
Analysis facilities: requirements

- **Interactivity**: AF needs to support both interactive and batch mode
- **Low latency data access**: AF are expected to require low latency random access media to achieve best performance (via data access patterns)
  - Required as a part of uncertainties for WAN IO needs at Tier-2 centers hosting AF
  - A trade off investments in disk space in caching infrastructure against network bandwidth use
- **Reusability**: AF should support extraction of user defined data formats to migrate onto laptops, desktops, workstations at home institutions or at home
- **Easy Deployment**: AF services expected to be deployed with industry standard platforms like Kubernetes and etc. to facilitate easy deployment within a Tier-3s
Analysis facilities: object storages

- Column-based data delivery services such as ServiceX, Coffea columnservice, or SkyHook will require object stores such as Ceph or Minio to be provided by the facility.

---

M. Weinberg, ServiceX: A columnar data delivery service for CMS

Arxiv 2007.01789
IRIS-HEP Future Analysis Systems and Facilities Workshop “brainstorming”
CMSAF @ UNL Setup

- JH setup: [https://github.com/CoffeaTeam/jhub](https://github.com/CoffeaTeam/jhub) (except specific secrets)
- Docker images for Dask Scheduler and Worker: [https://github.com/CoffeaTeam/coffea-casa](https://github.com/CoffeaTeam/coffea-casa)
  - [https://hub.docker.com/r/coffeateam/coffea-casa](https://hub.docker.com/r/coffeateam/coffea-casa)
  - [https://hub.docker.com/r/coffeateam/coffea-casa-analysis](https://hub.docker.com/r/coffeateam/coffea-casa-analysis)
- Docker image for JupyterHub (to get macaroons in the launch env)
  [https://github.com/clundst/jhubDocker](https://github.com/clundst/jhubDocker)
- Tutorials: [https://github.com/CoffeaTeam/coffea-casa-tutorials](https://github.com/CoffeaTeam/coffea-casa-tutorials)
JupyterHub + JupyterLab + Dask setup @ UNL

- JH is launched using Helm charts (together with users secrets)
CMSAF @ UNL Setup: Internals

- Docker image starting JupyterLab is integrated with HTCondor Dask Scheduler communicating with T3
  - Powered by Dask Labextension, which is integrated in the Docker image

Server Options

- Coffea Base Image
  - Coffea-casa build with coffea/dask/condor and cheese

- Minimal environment
  - To avoid too much bells and whistles: Python.

- Data science environment
  - If you want the additional bells and whistles: Python, R, and Julia.

- Spark environment
  - The Jupyter Stacks spark image

- Carl’s test image...here be dragons
  - Test environment

Image with integrated Dask scheduler and full coffea environment
Custom method CoffeaCasaCluster hidden in Dask Labextension (on top of 
dask_jobqueue.HTCondorCluster) to deploy Dask on common HTCondor job queue 
with possibility to specify Dask worker image and other parameters)
CMSAF @ UNL Analysis: Demo

```
[4]: # Generates a 1D histogram from the data output to the 'MET' key. fill_opts are optional, to fill the graph (default is a line).
     hist.plot1d(output['MET'], overlay='dataset', fill_opts={'edgecolor': (0,0,0.3), 'alpha': 0.8})

[4]: <matplotlib.axes._subplots.AxesSubplot at 0x7fd945413b50>
```

```
[5]: # Easy way to print all cutflow dict values. Can just do print(output['cutflow']["KEY_NAME"]) for one.
     for key, value in output['cutflow'].items():
         print(key, value)
	null events 53446198
	number of chunks 214
```
CMSAF @ UNL Analysis: Demo