Coffea-casa: an analysis facility prototype

Matous Adamec[1], *Garhan* Attebury[1], *Kenneth* Bloom[1], *Brian* Bockelman[2], *Carl* Lundstedt[1], *Oksana* Shadura[1], and *John* Thiltges[1]

[1]University of Nebraska-Lincoln, Lincoln, NE 68588
[2]Morgridge Institute for Research, 330 N. Orchard Street, Madison, WI 53715

Abstract. Data analysis in HEP has often relied on batch systems and event loops; users are given a non-interactive interface to computing resources and consider data event-by-event. The "Coffea-casa" prototype analysis facility is an effort to provide users with alternate mechanisms to access computing resources and enable new programming paradigms. Instead of the command-line interface and asynchronous batch access, a notebook-based web interface and interactive computing is provided. Instead of writing event loops, the column-based Coffea library is used.

In this paper, we describe the architectural components of the facility, the services offered to end users, and how it integrates into a larger ecosystem for data access and authentication.

1 Introduction

The ultimate purpose of computing for high-energy physics is to serve physicists in minimizing "time to insight". Physicists want quick access to data and incremental access to results, so that outputs can be examined almost as soon as the computational job is submitted. The tools for this must be approachable for inexperienced users while providing value to power users on the largest data analyses.

Traditionally, computing for physics data analysts has been delivered through "analysis facilities", which provide physicists with a login account that has access to disk storage hosting experiment datasets and a batch-processing system for user jobs that take the data as input. The LPC Central Analysis Facility [1] at Fermilab is an example that has served Compact Muon Solenoid (CMS) [2] physicists well during Run 1 and Run 2 of the Large Hadron Collider (LHC). It has over 11,000 batch slots available to users, who can access data over 10 PB of local storage. These facilities are typically used by physicists for a final round of data reduction, converting events in a 60 kB/event "MiniAOD" format [3] at 25 Hz to an analysis-specific 1 kB/event ntuple format that can later be processed at 1 kHz to make plots.

However, this model of an analysis facility – and of analysis as a computational task – will face challenges during the upcoming High-Luminosity LHC (HL-LHC) era. With 5 times increased trigger rates compared to recent experience, the number of events needed to be processed for an analysis will increase by about a factor of thirty. To keep the cost of storage under control, most analyses will have to switch from MiniAOD to a much smaller data format, a "NanoAOD" [4] of only 2 kB/event, but this format will likely be missing data elements necessary for any given analysis. New approaches to analysis facilities are thus increasingly urgent.

At the same time, new tools for data analysis have become available. The Python ecosystem provides a diverse suite of analysis tools that are in broad use in the scientific community [5]. Many scientists are taking advantage of Jupyter notebooks [6], which provide a straightforward way of documenting a data analysis. Column-wise data analysis [7], in which a single operation on a vector of events replaces calculations on individual events serially, is seen as a way to for the field to take advantage of vector processing units in modern CPUs, leading to significant speed-ups in throughput. Also, declarative programming paradigms [8] can make it simpler for physicists to intuitively code their analysis.

The challenges ahead and the new tools available have led us to develop a prototype analysis facility which we have named “Coffea-casa” [9]. The name is inspired by the Coffea framework [10] which is implemented using the Python language. Coffea and Python, with its package ecosystem and commodity big data technologies, improve time-to-insight, scalability, portability, and reproducibility of analysis. The Coffea-casa facility is built to natively support that framework.

In Coffea-casa, the user is presented with a Jupyter notebook interface that can be populated with code from a Git repository. When the notebook is executed, the processing automatically scales out to resources on the Nebraska CMS Tier-2 facility, giving the user transparent interactive access to a large computing resource. The facility has access to the entire CMS data set, thanks to the global data federation and local caches. An important feature is access to a “column service”; if a user is working with a compact data format (such as a NanoAOD) that is missing a data element that the user needs, the facility can be used to serve that “column” from a remote site. This allows only the compact data formats to be stored locally and augmented only as needed, a critical strategy for CMS to control the costs of storage in the HL-LHC era. Ultimately, Coffea-casa provides interactivity, low-latency data access, reusability, and easy deployment.

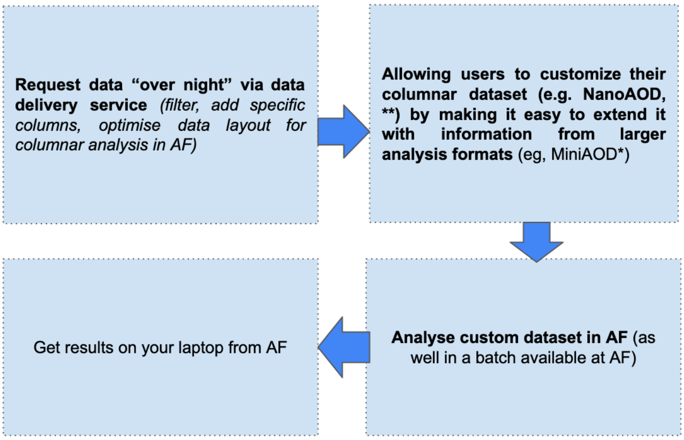
In this paper we describe the design of Coffea-casa and its deployment at Nebraska, early measurements of performance, and future directions.

2 Analysis Facility Motivation and Design

2.1 Analysis Facilities In Context

The Nebraska Tier-2 facility [11] has over 13,000 compute cores managed by HTCondor [12] along with 13 PB of data storage managed by HDFS; the facility is primarily for use by the CMS experiment. The facility’s resources are managed via a batch system that is only accessible through a grid-enabled gateway. The normal usage for the site is asynchronous; users could potentially wait hours for their first results. We desire to provide synchronous interfaces, such as web-based Jupyter notebooks, that have “instant” results. Unfortunately, “instant” is impossible given finite compute resources. Instead, we aim to provide instant *first* results through modest dedicated resources and have the analysis achieve scale by overlaying on top of the existing batch system.

Our eventual goal is to deliver the workflow described by Figure 1 for users. Users request custom data (on an “overnight” timescale) with filtering, custom columns, and an optimized data layout for columnar analysis; said data is made available via deployed data delivery services and the local storage. For the CMS experiment, a user may also extend existing NanoAOD data such as requesting info from the more comprehensive MiniAOD datasets. Users are subsequently able to access the data through the web browser and code custom analyses in the notebook.



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

Allowing users to customize their columnar dataset (e.g. NanoAOD, **) by making it easy to extend it with information from larger analysis formats (eg, MiniAOD*)

Analyse custom dataset in AF (as well in a batch available at AF)

Get results on your laptop from AF

Figure 1. The "ideal" user workflow that stimulated the design of an analysis facility.

2.2 The Coffea-casa Analysis Facility at Nebraska

As shown in Figure 2, the Coffea-casa prototype is built on a combination of Nebraska's multi-tenant Kubernetes [13] resource and HTCondor pool. Kubernetes is a service orchestration layer, allowing the administrator team and services to programmatically deploy cyberinfrastructure such as custom pods, network connectivity, and block devices. For example, Kubernetes hosts a JupyterHub [14] instance that authenticates users and, upon login, provisions a dedicated 8-core "analysis pod" for the user. A Kubernetes "pod" is the atomic unit of compute and service management (potentially consisting of several containers). For Coffea-casa, the analysis pod consists of a Jupyter notebook, a Dask [18] scheduler, and a Dask worker container. The Dask worker provides immediate execution of user tasks, provided the user with the feel of "instant" responsiveness of the system.

Kubernetes not only allows programmatic service deployment but provides "infrastructure as code"; the entire infrastructure can be expressed as static (YAML) files that can be shared and used by others. The facility utilizes a GitOps philosophy [15], keeping all files in a source code repository and allowing an agent (Flux [16]) to manage synchronization between the repository and the Kubernetes cluster. Multiple agents can deploy from the same repository, allowing separate production and development instances. Finally, the Coffea-casa team packages the core infrastructure (e.g., removing the site-specific passwords and secret keys) as a Helm chart [17], a format used by the Kubernetes community for application packaging.

2.3 Dask: Scalable Analytics in Python

A key component of Coffea-casa is the use of Dask. Dask [18] is a flexible library for parallel computing in Python. It builds on top of the Python ecosystem (e.g. Numpy [19], Pandas [20] and Scikit-Learn [21]) and is one of the backends of the Coffea library for distributing analyses tasks across multiple workers.

In addition to the parallelization, Coffea-casa heavily leverages four key features of Dask and its associated packages:

- Dask's data transport can be secured with TLS, allowing workers to securely join from separate networks and external clients to connect directly to the scheduler. We expose the scheduler to the Internet to allow users who prefer to connect directly to the Dask cluster

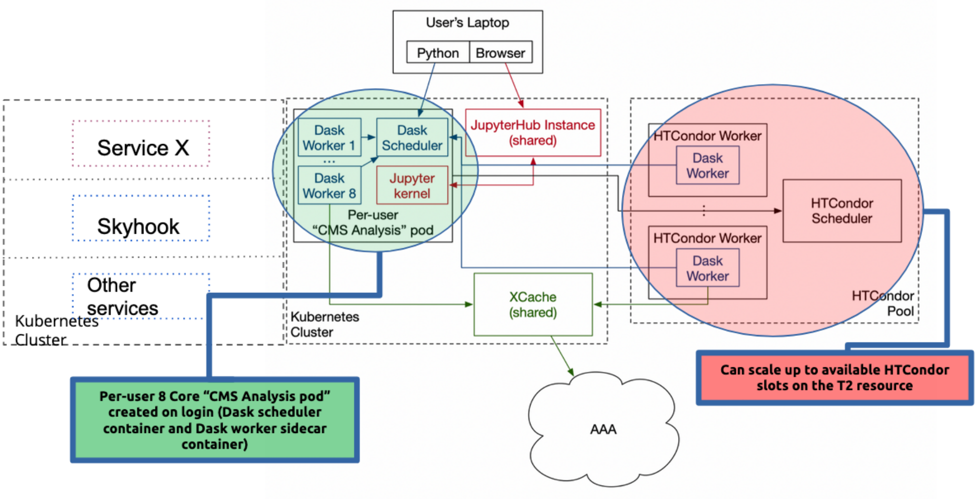


Figure 2. CMS analysis facility at the University of Nebraska. Note the Coffea-casa services in Kubernetes are able to integrate closely with the existing large-scale facility at Nebraska. Also shown are planned future services for data access and delivery, ServiceX and Skyhook.

from a Python session on their laptop instead of using the notebook-based interface. As Dask uses TLS, we can leverage SNI [22] to uniquely identify the requested hostname and utilize a TLS proxy such as Traefik [23] to route requests from outside the cluster to the correct Dask cluster. This provides a mechanism for all Dask clusters in Coffea-casa to share the same IP address and port instead of a unique IP per Dask scheduler.

- Dask can auto-scale, acquiring more resources based on the current task queue length and shutting down unnecessary workers when all tasks complete.
- The *dask-jobqueue* [24] package allows Dask to submit workers directly to a batch system; this feature provides users with resources from the much larger Nebraska computing facility instead of limiting them to the fixed-size analysis pod. Together with the dedicated worker, this provides an interactive experience using batch resources.
- The Dask Labextention [25] for JupyterLab integrates the Dask control into the JupyterLab web UI, providing easier management of the resources.

The resulting JupyterHub interface with Dask Labextention presented to users is shown in Figure 3.

The Dask worker and scheduler must share several Python libraries to function. Coffea-casa offers a customized analysis Docker container [26] to fill this need and, eventually, to provide this functionality to other sites.

To handle the customizations needed for the Coffea-casa environment, we developed the *CoffeaCasaCluster* object [27], an extension of Dask-jobqueue's *HTCondorCluster* object. For example, *CoffeaCasaCluster* ensures the Dask worker starts with the appropriate Docker container in the HTCondor batch system with the firewall ports configured correctly.

2.4 Authentication

An analysis facility needs to manage access to data and resources to the defined user community; however, at Nebraska, we strongly desire to avoid managing yet another set of user names and passwords, especially for a large experiment like CMS. The Coffea-casa facility uses OpenID Connect (OIDC) [28] to authenticate the users in browser using identities provided by a remote identity provider; in the current deploy, we use an IAM [29] instance at CERN. This allows users to utilize their CERN single-sign-on credentials to authenticate with the facility. IAM also provides some authorization management as the current instance verifies the user is registered with the CMS experiment in the CERN HR database.

The end result of this configuration is that any CMS user can utilize the facility, users don't need to manage new credentials, and we can guarantee *all* users are in CMS – without the intervention from Nebraska administrators. Once authenticated, we generate several local credentials tied to the CERN identity. These include:

- X.509 credentials (including a CA, host certificate, and user certificate) for use in Dask.
- A token for authentication with HTCondor, required for Dask scale-out to the larger local compute resources.
- A data access token for authentication with a local XRootD server.

The facility provides no mechanism for acquiring GSI / X.509 [30] credentials; this is a purposeful design decision aimed to avoid the complexities of moving credentials or managing their lifetimes.

2.5 Integration with Data Access

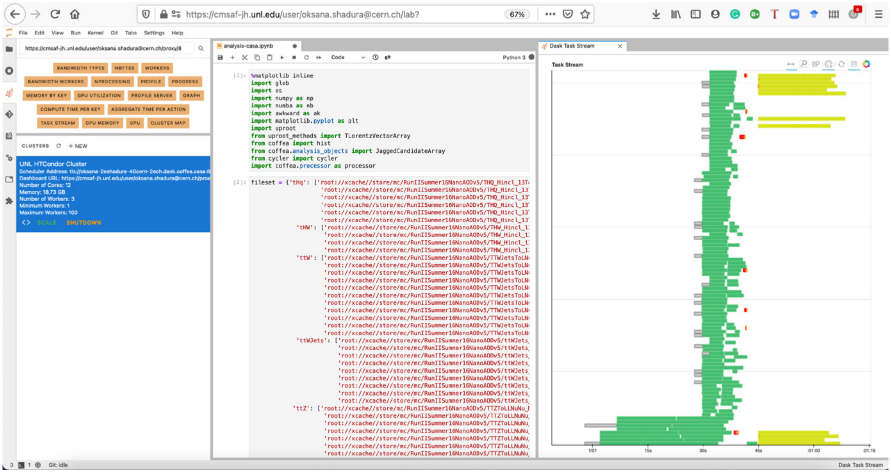
Coffea-casa users need to access data hosted by an HEP experiment; while these are traditionally secured with GSI, users do not have a GSI credential within the facility. Instead, the auto-generated data access token can be used to authenticate with a proxy service based on XRootD/XCache [31]. The proxy service, operated by the facility administrators, has a valid credential for accessing experiment data. Data is downloaded on demand from an infrastructure like CMS's "Any Data, Any Time, Anywhere" (AAA) data federation [32]. In the current deployment, because we can guarantee all authenticated users are valid CMS members, we assume any authenticated access to the proxy service is allowed to access CMS data. We envision that each experiment will have their separate portal and their separate credentials.

Finally, to ease the discovery of the correct internal hostname of the proxy and setup of the token, we provide a custom plugin to the XRootD client [33]. This plugin intercepts all XRootD file-open requests, changes the URL to point at the local proxy, and adds the data access token to the session authentication.

3 Coffea-casa Performance on a Sample Analysis

To demonstrate the capabilities of Coffea-casa in practice, we took two coffea-based analysis from CMS and scaled them in the facility. The two sample analyses are the production of four top quarks (FTA) [34], which processes of 740 million events across 1377 datasets, and single top-Higgs production (tHQ) [35], which processes 116 million events across 78 datasets. Input files are in the NanoAOD format and are cached locally in the XCache endpoint. Both analysis are implemented in the Coffea framework in Jupyter notebook format and executed from within Coffea-casa.

We examined the effect of the number of workers on the average of the observed event processing rates; results are shown in Figure 4. We fixed the number of workers to remove the side effects of cluster scheduling on the measurement. The observed event throughput increases until it peaks at about 350 kHz for 50 worker nodes for the FTA analysis. After that, we believe the benefits of increased workers are outweighed by factors such as start-up



UNL HTCondor Cluster
Scheduler Address: tls://oksana-2eshadura-40cern-2ech.dask.coffea.casa:8
Dashboard URL: https://cmsaf-jh.unl.edu/user/oksana.shadura@cern.ch/prox
Number of Cores: 12
Memory: 18.73 GB
Number of Workers: 3
Minimum Workers: 1
Maximum Workers: 100

```
[1]: %matplotlib inline
     import glob
     import os
     import numpy as np
     import numba as nb
     import awkward as ak
     import matplotlib.pyplot as plt
     import uproot
     from uproot_methods import TLorentzVectorArray
     from coffea import hist
     from coffea.analysis_objects import JaggedCandidateArray
     from cycler import cycler
     import coffea.processor as processor

[2]: fileset = {'tHq': ['root://xcache//store/mc/RunIISummer16NanoAODv5/THQ_Hincl_13T
                       'root://xcache//store/mc/RunIISummer16NanoAODv5/THQ_Hincl_13
                       'root://xcache//store/mc/RunIISummer16NanoAODv5/THQ_Hincl_13
                       'root://xcache//store/mc/RunIISummer16NanoAODv5/THQ_Hincl_13
                       'root://xcache//store/mc/RunIISummer16NanoAODv5/THQ_Hincl_13
                       'root://xcache//store/mc/RunIISummer16NanoAODv5/THQ_Hincl_13
                'tHW': ['root://xcache//store/mc/RunIISummer16NanoAODv5/THW_Hincl_1
                        'root://xcache//store/mc/RunIISummer16NanoAODv5/THW_Hincl_1
                        'root://xcache//store/mc/RunIISummer16NanoAODv5/THW_Hincl_1
                        'root://xcache//store/mc/RunIISummer16NanoAODv5/THW_Hincl_1
                'ttW': ['root://xcache//store/mc/RunIISummer16NanoAODv5/TTWJetsToLN
                        'root://xcache//store/mc/RunIISummer16NanoAODv5/TTWJetsToLN
                        'root://xcache//store/mc/RunIISummer16NanoAODv5/TTWJetsToLN
                        'root://xcache//store/mc/RunIISummer16NanoAODv5/TTWJetsToLN
                        'root://xcache//store/mc/RunIISummer16NanoAODv5/TTWJetsToLN
                        'root://xcache//store/mc/RunIISummer16NanoAODv5/TTWJetsToLN
                        'root://xcache//store/mc/RunIISummer16NanoAODv5/TTWJetsToLN
                        'root://xcache//store/mc/RunIISummer16NanoAODv5/TTWJetsToLN
                        'root://xcache//store/mc/RunIISummer16NanoAODv5/TTWJetsToLN
                        'root://xcache//store/mc/RunIISummer16NanoAODv5/TTWJetsToLN
                        'root://xcache//store/mc/RunIISummer16NanoAODv5/TTWJetsToLN
                'ttWJets': ['root://xcache//store/mc/RunIISummer16NanoAODv5/ttWJets_
                            'root://xcache//store/mc/RunIISummer16NanoAODv5/ttWJets_
                            'root://xcache//store/mc/RunIISummer16NanoAODv5/ttWJets_
                            'root://xcache//store/mc/RunIISummer16NanoAODv5/ttWJets_
                            'root://xcache//store/mc/RunIISummer16NanoAODv5/ttWJets_
                            'root://xcache//store/mc/RunIISummer16NanoAODv5/ttWJets_
                            'root://xcache//store/mc/RunIISummer16NanoAODv5/ttWJets_
                            'root://xcache//store/mc/RunIISummer16NanoAODv5/ttWJets_
                            'root://xcache//store/mc/RunIISummer16NanoAODv5/ttWJets_
               'ttZ': ['root://xcache//store/mc/RunIISummer16NanoAODv5/TTZToLLNuNu_
                       'root://xcache//store/mc/RunIISummer16NanoAODv5/TTZToLLNuNu_
                       'root://xcache//store/mc/RunIISummer16NanoAODv5/TTZToLLNuNu_
                       'root://xcache//store/mc/RunIISummer16NanoAODv5/TTZToLLNuNu_
                       'root://xcache//store/mc/RunIISummer16NanoAODv5/TTZToLLNuNu_
                       'root://xcache//store/mc/RunIISummer16NanoAODv5/TTZToLLNuNu_
                       'root://xcache//store/mc/RunIISummer16NanoAODv5/TTZToLLNuNu_
```

Figure 3. User interface for the Coffea-casa analysis facility, with Dask Labextension display (left), Jupyter notebook (middle) and Dask task stream monitoring (right).

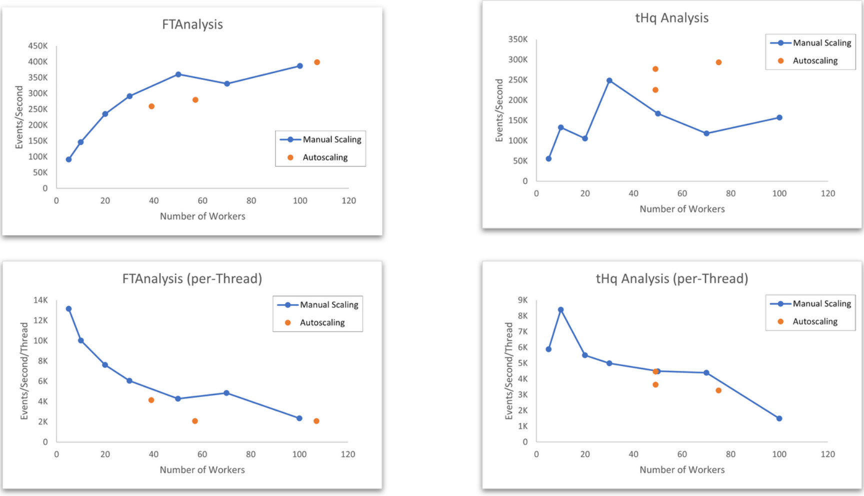


Figure 4. Results of from two different physics analyses, FTA (left) and tHq (right), with various numbers of workers each having 4 CPUs and 6 GB RAM memory.

times. While FTA analysis was tested with automatic scaling, where we couldn't predict strictly the number of workers (see the difference between manual scaling and auto scaling data points on Figure 4), from plot it was noticeable that ratio of events/second is slightly lower for the approximately the same amount of workers used with manual scaling. Instead, for tHQ analysis autoscaling actually showed better results. While performance testing is still a work in progress, we are going to investigate a reason of such different behaviour using featured by Coffea-casa various troubleshooting tools, such as a Dask performance profiling server to aid in investigation. A brief examination of the profiling reports for FTA analysis, available by default in Coffea-casa appears to show that there is a stall between the start and the first task on the worker in the task stream, which gets longer when adding more workers, as well as a performance difference between using Dask directly or executing analysis using the Dask Labextention cluster.

4 Conclusions and Future Work

The prototype analysis facility at Nebraska, Coffea-casa, serves as an effective prototype and demonstration of several technologies under development for use in HL-LHC analysis. However, significant work remains for it to be an everyday workhorse for users. For example, the scaling limits in Section 3 need to be better understood; we currently think the limit is the performance of the caching proxy hardware. Users need the ability to develop and (securely!) deploy their own custom containers for analysis; potentially, solutions like BinderHub [36] may be needed for the facility.

We believe an critical future feature will be access to a "column service"; if a user is working with a compact data format (such as a NanoAOD) that is missing a data element that the user needs, the facility can be used to serve that "column" from a remote site. This allows only the compact data formats to be stored locally and augmented only as needed, a critical strategy for CMS to control the costs of storage in the HL-LHC era. Data delivery services reduce local storage needs at analysis facilities by filtering and projecting input data (both row- and column-wise) and caching results, removing the need for manual bookkeeping and intermediate data storage by analysts.

As one of future items we plan to investigate ServiceX service which provides user-level ntuple production by converting experiment-specific datasets to columns and extracts data from flat ROOT [39] files. The service enables the simple cuts or simply derived columns, as well as specified fields. Other item in Coffea-casa to-do list is to investigate Skyhook DM project that can convert ROOT files, such as the NanoAOD format, to the internal object-store format. Instead of providing a file-like interface, the project is working to implement Ceph [40] APIs to natively filter and project structured data, delivering it to Dask workers through the Arrow format. This object API is expected to eventually offer "join" functionality, allowing users to include their bespoke columns into a shared dataset.

While work remains, Coffea-casa demonstrates features such as efficient data access services, notebook interfaces, token authentication, and automatic, external cluster scaling. Initial users have been testing the facility to provide feedback and initial use cases and the team plans to distribute Coffea-casa products artifacts (such as Helm charts) for use at other Kubernetes-based sites.

5 Acknowledgements

This work was supported by the National Science Foundation under Cooperative Agreements OAC-1836650 and PHY-1624356 and awards PHY-1913513 and OAC-1450323.

References

[1] Bloch, I. *The LHC Physics Center*. Nuclear Physics B-Proceedings Supplements, **177**, pp.261-262, Elsevier, 2008.

[2] Adolphi, Roman, et al. *The CMS experiment at the CERN LHC*. Jinst, **803**: S08004, 2008.

[3] Petrucciani, Giovanni, et al. *Mini-AOD: A new analysis data format for CMS.* Journal of Physics: Conference Series. **Vol. 664.** No. 7. IOP Publishing, 2015.

[4] Rizzi, Andrea, Giovanni Petrucciani, and Marco Peruzzi. *A further reduction in CMS event data for analysis: the NANOAOD format.*, EPJ Web of Conferences. **Vol. 214.**, EDP Sciences, 2019.

[5] Travis E. Oliphant. *Python for Scientific Computing, Computing in Science & Engineering*, 9, 10-20 (2007), DOI:10.1109/MCSE.2007.58

[6] Thomas Kluyver, et.al., *Jupyter Notebooks – a publishing format for reproducible computational workflows*, DOI:10.3233/978-1-61499-649-1-87

[7] J Pivarski, P Elmer, D Lange, *Awkward Arrays in Python, C++, and Numba*, arXiv:2001.06307

[8] Danilo Piparo, Philippe Canal, Enrico Guiraud, Xavier Valls Pla, Gerardo Ganis, Guilherme Amadio, Axel Naumann and Enric Tejedor *RDataFrame: Easy Parallel ROOT Analysis at 100 Threads* EPJ Web Conf., 214 (2019) 06029 DOI: https://doi.org/10.1051/epjconf/201921406029

[9] Shadura, Oksana. (2020, July). *A prototype U.S. CMS analysis facility*. Presented at the PyHEP 2020 Workshop, Zenodo. http://doi.org/10.5281/zenodo.4136273

[10] Nicholas Smith, Lindsey Gray, Matteo Cremonesi, Bo Jayatilaka, Oliver Gutsche, Allison Hall, Kevin Pedro, Maria Acosta, Andrew Melo, Stefano Belforte and Jim Pivarski, *Coffea Columnar Object Framework For Effective Analysis*, EPJ Web Conf., **245**, 06012 (2020)

[11] K Bloom, *US CMS Tier-2 computing* 2008 J. Phys.: Conf. Ser. 119 052004

[12] Douglas Thain, Todd Tannenbaum, and Miron Livny, "Distributed Computing in Practice: The Condor Experience", *Concurrency and Computation: Practice and Experience*, Vol. 17, No. 2-4, pages 323-356, February-April, 2005.

[13] Kelsey Hightower, Brendan Burns, and Joe Beda. *Kubernetes: Up and Running Dive into the Future of Infrastructure* (1st. ed.). O'Reilly Media, Inc. 2017

[14] www.jupyter.org. (n.d.). Project Jupyter. [online] Available at: https://jupyter.org/hub.

[15] Atlassian (n.d.). Is GitOps the next big thing in DevOps? | Atlassian Git Tutorial. [online] Atlassian. Available at: https://www.atlassian.com/git/tutorials/gitops [Accessed 28 Feb. 2021].

[16] GitHub. (2021). fluxcd/flux. [online] Available at: https://github.com/fluxcd/flux [Accessed 28 Feb. 2021].

[17] Helm.sh. (2019). Helm. [online] Available at: https://helm.sh/ [Accessed 28 Oct. 2019].

[18] Rocklin, Matthew. *Dask: Parallel computation with blocked algorithms and task scheduling.*, Proceedings of the 14th python in science conference. **Vol. 126.**, Austin, TX: SciPy, 2015.

[19] Van Der Walt, Stefan, S. Chris Colbert, and Gael Varoquaux. *The NumPy array: a structure for efficient numerical computation.* Computing in science & engineering 13.2 (2011): 22-30.

[20] McKinney, Wes. *pandas: a foundational Python library for data analysis and statistics.* Python for High Performance and Scientific Computing 14.9 (2011): 1-9.

[21] Pedregosa, Fabian, et al. *Scikit-learn: Machine learning in Python.* the Journal of machine Learning research 12 (2011): 2825-2830.

[22] Eastlake, Donald. *Transport layer security (TLS) extensions: Extension definitions*. RFC 6066, January, 2011.

[23] Traefik Labs: Makes Networking Boring. (n.d.). *Traefik Labs: Makes Networking Boring.* [online] Available at: https://traefik.io/., accessed: 2021-02-25

[24] GitHub. (2021). dask/dask-jobqueue. [online] Available at: https://github.com/dask/dask-jobqueue [Accessed 28 Feb. 2021].

[25] GitHub. (2021). dask/dask-labextension. [online] Available at: https://github.com/dask/dask-labextension [Accessed 28 Feb. 2021].

[26] hub.docker.com. (n.d.). Docker Hub. [online] Available at: https://hub.docker.com/u/coffeateam [Accessed 28 Feb. 2021].

[27] GitHub. (2021). CoffeaTeam/coffea-casa. [online] Available at: https://github.com/CoffeaTeam/coffea-casa [Accessed 28 Feb. 2021].

[28] Sakimura, Natsuhiko, et al. *Openid connect core 1.0.*, The OpenID Foundation (2014): S3.

[29] cms-auth.web.cern.ch. (n.d.). INDIGO IAM for cms-Log in. [online] Available at: https://cms-auth.web.cern.ch [Accessed 28 Feb. 2021].

[30] Welch, Von, et al. *X. 509 proxy certificates for dynamic delegation.* 3rd annual PKI R&D workshop. Vol. **14**. 2004.

[31] Fajardo, Edgar, et al. *Creating a content delivery network for general science on the internet backbone using XCaches.*, EPJ Web of Conferences. **Vol. 245.**, EDP Sciences, 2020.

[32] K. Bloom, et al., *Any data, any time, anywhere: Global data access for science*, in 2015 IEEE/ACM 2nd International Symposium on Big Data Computing (BDC), pp. 85–91. 2015. doi:10.1109/BDC.2015.33.

[33] Bockelman, B.P. (2020). bbockelm/xrdcl-authz-plugin. [online] GitHub. Available at: https://github.com/bbockelm/xrdcl-authz-plugin [Accessed 28 Feb. 2021].

[34] CMS Collaboration, *Search for the production of four top quarks at the CMS experiment at* $\sqrt{s}$*= 13 TeV*, JHEP **11** (2019) 082.

[35] CMS Collaboration, *Search for associated production of a Higgs boson and a single top quark in proton-proton collisions at* $\sqrt{s}$*= 13 TeV*. Physical Review D, **99(9)**, 092005 (2019).

[36] GitHub. (2021). jupyterhub/binderhub. [online] Available at: https://github.com/jupyterhub/binderhub [Accessed 28 Feb. 2021].

[37] Galewsky, B., et al. *ServiceX A Distributed, Caching, Columnar Data Delivery Service.*, EPJ Web of Conferences. **Vol. 245.**, EDP Sciences, 2020.

[38] Jeff LeFevre, Carlos Maltzahn. *SkyhookDM: Data Processing in Ceph with Programmable Storage.*, retrieved from https://par.nsf.gov/biblio/10182302. USENIX login; 45.2, accessed: 2021-02-25

[39] Rene Brun and Fons Rademakers, *ROOT - An Object Oriented Data Analysis Framework*, Proceedings AIHENP 96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. **A 389** (1997) 81-86.

[40] Weil, Sage A., et al. *Ceph: A scalable, high-performance distributed file system.* Proceedings of the 7th symposium on Operating systems design and implementation. 2006.