Snowmass Letter of Interest - Analysis Facilities - CompF5

Burt Holzman <burt@fnal.gov> (Fermilab), Lindsey Gray (Fermilab), Bo Jayatilaka (Fermilab), Nick Smith (Fermilab)

Overview

In this letter, we comment on the “last mile” for computing in support of physics analyses; this does not cover simulation/acquisition, batch processing/production/reconstruction, and data reduction. Specifically, we focus on analysis facilities, which provide the foundation for the individual researcher to experiment with and understand their data.

User Experience

Recently, analysis in high energy physics has undergone significant changes. Analyses were written in C++; they are now being written in Python. Many analysis scripts are now executed and published via Jupyter notebooks[1] rather than the Linux shell. In an analysis facility, as a primary interface for analyzers, we will provide access to Jupyter notebooks “as a service” using JupyterHub. As a secondary interface, we will continue to provide shell access. Authentication and authorization will be performed using a federated identity model (subject to the security constraints of the ecosystem where the facility is sited). The need for computation on non-CPU architectures (e.g. GPUs) while still maintaining the fast “time-to-insight” required for exploratory analysis is growing as the average analysis dataset size grows. Batch job queues operate on a non-interactive timescale, whereas emerging technologies can provide sub-minute turnaround for tasks that are too large to fit on a single compute node. Moreover, a clear and approachable system for probing the structure and content of the data in real-time vastly improves the understanding and efficiency for analyzers, significantly lowering the bar for new students, and even experienced physicists switching experiments.

Underlying infrastructure

Services in the facility are deployed on a container-based infrastructure. Containers provide flexibility, portability and isolation without the additional overhead of virtual machines. Sites that deploy this infrastructure at a wide scale make it easier to add elasticity to the analysis facility; servers for a different purpose (e.g. batch worker nodes) can be repurposed on the fly for scheduling analyses. The orchestration tool of choice for containers is Kubernetes. It provides a unified description and configuration language, configuration management, service discovery and load balancing, automated rollouts and rollbacks, and other features key to providing stable services. It was originally designed for cloud computing, which adopts a single-tenant model:

[1] "Jupyter Notebook." https://jupyter.org/. Accessed 27 Aug. 2020.

This manuscript has been authored by Fermi Research Alliance, LLC under Contract No. DE-AC02-07CH11359 with the U.S. Department of Energy, Office of Science, Office of High Energy Physics.

one user creates and owns an entire cluster. We have adopted Red Hat's open-source OKD[2] platform on top of Kubernetes. OKD is designed for multi-tenancy - it includes additional security and isolation, adding operations-centric tools, a user-friendly GUI, and additional storage and network orchestration components.

Despite advances in network capacity, data locality is still relevant to the performance of analysis applications. Often, only a small fraction of data stored in traditional formats such as ROOT files needs to be accessed for a given operation. Services such as XCache can locally cache the specific byte ranges. Modern column-based data delivery services such as ServiceX, Coffea columnservice, or SkyHook would allow data access at column granularity rather than the current "data tier" (a fixed set of columns) granularity. These services require object stores such as Ceph or Minio to be provided by the facility.

As Kubernetes and related applications were designed in a single-tenant model, the current state-of-the-art for scheduling on these systems is primitive. We will integrate high throughput computing scheduling with the facility - from integrating these capabilities natively to dynamically provisioning a worker pool via HTCondor.

Analysis as application: traditional and low-latency

Traditional analyses in HEP generally consist of "row-wise" analyses. Simply put, these workflows serially process data from one event at a time, calculating derived quantities (e.g. invariant masses), aggregating these data into additional data structures (histograms, n-tuples, etc). These data are further aggregated and additional calculations are performed. The facility should support this mode of analysis.

There is a growing interest in the field in low-latency parallel "column-wise" analyses (although there is a long history of parallel processing frameworks[3]). Industry-standard tools for "big/tidy data" such as Apache Spark have been adopted. Given the widespread use of python, the Dask python parallel computing library is particularly attractive. As HEP and industry have significantly different data structures, the use of these frameworks requires additional

[2] "OKD - The Community Distribution of Kubernetes that powers ...." https://www.okd.io/. Accessed 27 Aug. 2020.

[3] D. Feichtinger *et al*., "PROOF - The Parallel ROOT Facility," *2006 15th IEEE International Conference on High Performance Distributed Computing*, Paris, 2006, pp. 379-380, doi: 10.1109/HPDC.2006.1652193.

middleware, such as uproot[4], AwkwardArray[5], and Coffea[6]. The facility should support this method of analysis as well.

[4] "scikit-hep/uproot: ROOT I/O in pure Python and NumPy. - GitHub." https://github.com/scikit-hep/uproot. Accessed 27 Aug. 2020.
[5] "Awkward Array | Institute for Research and Innovation in ...." https://iris-hep.org/projects/awkward.html. Accessed 27 Aug. 2020.
[6] "CoffeaTeam/coffea: Basic tools and wrappers for ... - GitHub." https://github.com/CoffeaTeam/coffea. Accessed 27 Aug. 2020.