Advancing Machine Learning Inference with Columnar Analysis at CMS Analysis Facilities

The high luminosity LHC (HL-LHC) era will offer incredible physics discovery opportunities with the delivery of luminosity well beyond the original design of the machine and upgraded CMS detectors with improved granularity. These hardware advances will also usher in unprecedented computing challenges to store, process, and analyze the data. At the same time, rapid advances in machine learning technology have led to an explosion in use of the computationally expensive techniques from event reconstruction, to particle identification, and data analysis. The collision of these two effects demands that a new paradigm is needed for widespread, adaptable, and efficient use of machine learning techniques in CMS analysis to achieve the maximum possible physics performance in a computing resource constrained environment. In this proposal, we seek support for a postdoctoral research associate at the University of Colorado Boulder, Alexx Perloff, to develop the tools required for rapid machine learning inference within the elastic analysis facility model being developed for next generation CMS analysis computing.

1 Introduction

CMS computing challenges are growing exponentially more difficult. Coupled with the breakdown of Dennard scaling somewhere in 2005-2007 and resulting inability to increase central processing unit (CPU) clock frequencies, by the time of the HL-LHC a completely new paradigm will be needed to maintain reasonable performance. In order to cope with these trends, CMS is tackling the problem from several angles.

As CPUs will not be able to handle the increased computing needs, more attention is being paid to highly parallelized workflows. A lot of effort has gone into writing programs for multicore CPUs and more recently into software taking advantage of general-purpose graphics processing units (GPUs). Based on past successes, CMS is aggressively pursuing a *heterogeneous* computing model, whereby some of the computation is offloaded from the CPU onto various accelerator components, like GPUs, FPGAs, and TPUs. Nevertheless, efficient and scalable access to these so called co-processors remains an open area of research.

The use of GPU co-processors within the CMS computing architecture has also been spurred on by increased interest and use in deep learning versus more traditional rule-based algorithms. At its core, the computations which are the backbone of deep learning use simple, but large-scale matrix multiplications, which can be highly parallelized and offloaded onto the GPUs. While there continue to be improvements to the more traditional algorithms, machine learning (ML) based algorithms are often more powerful than their rule-based counterparts and offer predictions much faster. Additionally, it's often the case that ML versions are more portable between computing architectures, for example systems with various types of processors or co-processors.

In CMS, there are already several projects aiming at replacing core event reconstruction algorithms (i.e. tracking, calorimetry, etc.), object reconstruction algorithms (i.e. particle flow), and object ID algorithms (i.e. jet tagging, tau ID) with ML counterparts. As the percentage of ML algorithms grows within the CMS reconstruction workflow, offloading the inferencing stage of these ML algorithms to a co-processor will become even more important. However, as it currently stands the ML inferencing is in the shadow of the many other components of the production and reconstruction workflow. In the analysis workflow, on the other hand, the *time to insight* can often be completely dominated by the ML algorithms within an event loop. The ability to offload the ML component of an analysis to a co-processor and speed up the time to inference would be a huge benefit.

Based on the stated benefits of ML algorithms and co-processors, it would be simplest to have a tightly coupled heterogeneous computing system where there was a fixed ratio of co-processors to CPUs. However, this is both incredibly wasteful, with the co-processor sitting idle for large portions of time, and extremely expensive. Instead, the true ratio is dependent on the type of workflow – MC generation, reconstruction, analysis, ML training, etc. – and will change over time as workflows and algorithms develop. This necessitates that a system be developed whereby the main CPU process makes calls to a remote accelerator, allowing for a variable number of co-processors to be dedicated to a given workflow.

Another ongoing change within the landscape of CMS analysis is the push toward using columnar analysis tools. These tools, often Python-based, provide several benefits which, if properly utilized, could speed up the time to insight. Historically, CMS analyses have used an "event loop" pattern to access data, processing one event's worth of information per loop iteration. While this seems logical since each CMS event is independent, the data is actually stored, not event-by-event, but with individual objects or properties sequentially stored in memory. Using an event loop forces the computer to read information from non-contiguous portions of memory. Columnar analyses, by contrast, access whole blocks (columns) of memory and only those columns, reducing the overall I/O overhead, and then perform a given operation on the entire column. Packages like Coffea [1] seek to make columnar analyses more accessible and easier to use by packaging all of the necessary tools into a single bundle and including platform independent facilities for horizontal scaling when computing problems become large.

In this proposal we will show a plan to build upon the existing columnar analysis tools and to be able to perform rapid, interactive ML inferencing by utilizing calls to a remote cluster of GPU co-processors where resources can scale as needed. Such a use case would allow for an efficient use of shared resources as well as a potentially dramatic reduction in the time to insight for physics analyses.

2 Proposal

This proposal seeks 50% remuneration for a postdoc to study the use of machine learning within a columnar analysis framework running on the Fermilab Elastic Analysis Facility (EAF) [2, 3]. The EAF is a heterogeneous multi-tenant Kubernetes cluster where users are able to scale out their columnar analysis workflows using Coffea [1] and Dask [4]. The project was originally prototyped in 2018 and then redeployed in 2021 using the experience gained over the previous 3 years. In addition to the resources at the EAF, including GPUs, users are able to take advantage of the more than 20,000 CPU cores available on the Fermilab batch computing farms.

The goal of the project would be to combine the benefits of several of the aforementioned technologies:

- The higher throughput of columnar analysis tools.
- The possibility to have higher sensitivity, lower latency, and greater portability using machine learning.

- The ability to speed up the time to inference of ML using co-processors remotely accessed through calls to Nvidia Triton [5] servers already setup at the FNAL EAF.
- Taking advantage of the natural ability of a columnar analysis to fill the GPU registers with many events worth of input features without the need to aggregate information from several streams.
- The resources of an elastic analysis facility where much of the complication of scaling out an analysis and accessing co-processors is abstracted away.

While the various pieces of hardware, the computing environment, and tools have theoretically been setup at the EAF, more work needs to be done to put all of the pieces together. Additionally, work should be done to see if anything is missing to perform analyses at scale. The goal here is to have a "production" ready system that multiple users can use simultaneously. The system must be tested to make sure that it is stable not just for one user, but many.

The postdoc will first choose and develop a representative columnar physics analysis. This analysis framework will serve as a testing ground for including the machine learning inferencing as well as for testing the workflow at various scales. Complementing the R&D activities proposed here, Perloff has worked as an early adopter of columnar tools working with Fermilab scientist Kevin Pedro to adapt their emerging jets analysis to run in the Coffea framework. Much of the work was completed using interactive Jupyter, with the heavy computation jobs being done using Dask on the LPC batch cluster, just as we would envision performing analyses using the Fermilab EAF. While the current iteration of the analysis doesn't involve machine learning, Perloff has already started work on a future iteration which relies heavily on a graph neural network (GNN) based jet tagger. It's this framework which will be used to test and benchmark the system at various scales and with varying numbers of users.

After a suitable analysis test bed has been developed and characterized, the postdoc will then curate a set of benchmark ML models on which to test the inferencing capabilities of the EAF. A single model may be used to test and benchmark a homogeneous workload for a single analysis at various scales. However, we know that analysis workflows won't be homogeneous. The additional benchmark models will be used to stress test the system testing analyses with multiple inference needs (multiple models) and multiple analyses making requests at the same time, each using a different model. This will be a true test of the partially existing load balance, which keeps track of the Triton Servers hosting a given model. The goal is to test that an inference request can be sent and predictions received reliably.

Another synergistic tie-in with current work by the CU group is the use of a GNN-based emerging jet tagger being developed by graduate student Claire Savard in collaboration with Perloff and Ulmer. This is one such model which would make sense to interface with the proposed analysis framework and would provide a realistic test of analysis needs.

Following the development of a suitable analysis platform described above, the postdoc will move onto benchmarking the performance of the analysis code and comparing it to the inference throughput performed on a local GPU. The Nvidia Triton servers running in the Fermilab EAF are visible from the LPC cluster or Dask [4] worker nodes. The goal here is to see that the worker nodes can effectively talk to the GPU and make an inference request. Using this as a basis of comparison the postdoc will then tackle multi-user scale-out issues and model heterogeneity. In doing this there is no predicting what problems will be encountered; one can't expect a couple hundred GPUs to work when many users and/or worker nodes make requests at the same time. Along these lines, there are several questions that need to be answered:

- While the GPUs are available through Kubernetes pods and resources are dealt out as the service needs to scale, how do we measure and determine when the system should scale up or down?
- How should resources be partitioned for multiple users?
- What are the failure modes of the system?
- How much fault tolerance can be built into the system?
- What should the system/users know about where their models are located and on which GPUs? How will model versioning be handled?

Assuming all of these answers can be gathered during the first year, a stretch goal for the project would be to try to reproduce a working setup at another GPU equipped analysis facility, like the Coffea-casa facility at University of Nebraska Lincoln.

The proposed research project is in line with the vision of several groups within the CMS collaboration. The CMS Machine Learning group has a vested interest in making fast inferencing of machine learning models accessible to more analyzers, including those using state of the art tools and facilities. The project is also aligned with the interests of the US-CMS Software and Computing Operations and R&D efforts, especially with respect to columnar analyses performed at the LPC EAF and similar facilities.

The **deliverables** from this research would be a document describing the testing procedures and their results, listing any recommendations for changes to be implemented to the Fermilab EAF, and providing a set of easy to follow instructions for future analyzers.

2.1 Milestones

Milestones are planned to occur approximately each quarter. These milestones, listed below, are assumed to begin from the postdoc's start date.

- Months 1-3: Familiarization with existing Fermilab Elastic Analysis Facility architecture and software stack. Development of analysis code for use within the EAF to send an inference request and receive the predictions. Benchmark this analysis code, sans-GPU inferencing, to serve as a basis for comparison.
- Months 4-6: Development or modification of a suitable ML model(s) to benchmark performance within the analysis framework. Demonstrate the ability to send input data to a GPU and receive a prediction.
- Months 7-9: Benchmark the performance of the analysis code and compare it to inferencing on a local GPU. Stretch goal: Also compare to inferencing without using a columnar analysis or within a CMSSW environment using Services for Optimized Network Inference on Coprocessors (SONIC) [6].
- Months 10-12: Scale testing to simulate multiple users and models. Present results at international HEP Computing meetings/workshops. Stretch goal: Work with other EAF-like sites (i.e. Coffea-casa at T2_US_Nebraska or at ACCRE at T3_US_Vanderbilt) to try to reproduce a working setup.

2.2 Mentorship, Supervision, and Community Interaction

The postdoc will be supervised by PI Keith Ulmer. Having joined the CU CMS group in 2018, Perloff is already very well integrated into the CU CMS team with weekly group meetings and frequent discussions with faculty and junior members of the group, which will extend to include the proposed project. This mentoring includes a focus on professional growth and development. Participation in international meetings and workshops in computing and machine learning will be encouraged and supported as will taking visible leadership roles in these communities, such as Perloff's current L3 position in the CMS ML group.

Perloff will be based at the LPC during the project, which will facilitate continued engagement by the postdoc in the LPC S&C community. The postdoc has long had contact with the computing experts at the LPC, such as Lindsey Gray (FNAL), Michael Hildreth (Notre Dame), and Burt Holzman (FNAL), and will continue to draw upon those relationships when there are shared machine learning efforts. The postdoc and the PI will also continue to develop other relationships within the broader CMS Machine Learning community, including to the Coffea development team and the other personnel working on the FNAL Elastic Analysis Facility. Being located at the LPC will also enable the postdoc to engage the wider US CMS community about the benefits of ML inferencing with a columnar analysis environment.

3 Conclusion

With the upgrade to the High Luminosity LHC, the demands for offline computing resources in CMS will grow immensely. Simultaneously, advances in machine learning are rapidly expanding the roles that these powerful techniques can play, even further increasing the need for efficient use of computing power within the constrained environment. Our proposal seeks support for a postdoctoral researcher to develop fast and user friendly tools for the broad deployment of machine learning inference within the Elastic Analysis Facility environment.

References

[1] Lindsey Gray et al. *CoffeaTeam/coffea: Release v0.7.15.* Version v0.7.15. May 2022. DOI: 10.5281/zenodo.6555201. URL: https://doi.org/10.5281/zenodo.6555201.

[2] Maria Acosta Flechas et al. *Collaborative Computing Support for Analysis Facilities Exploiting Software as Infrastructure Techniques.* 2022. DOI: 10.48550/ARXIV.2203.10161. URL: https://arxiv.org/abs/2203.10161.

[3] Doug Benjamin et al. *Analysis Facilities for HL-LHC.* 2022. DOI: 10.48550/ARXIV.2203.08010. URL: https://arxiv.org/abs/2203.08010.

[4] Dask Development Team. *Dask: Library for dynamic task scheduling.* 2016. URL: https://dask.org.

[5] Nvidia. *Triton inference server, [software] version v2.21.0 (accessed 2022-05-16).* 2019. URL: https://docs.nvidia.com/deeplearning/sdk/triton-inference-server-guide/docs/index.html.

[6] Kevin Pedro. *SonicCMS.* Version v5.2.0. (accessed 18 May 2020). URL: https://github.com/fastmachinelearning/SonicCMS.