2021 3rd Annual Workshop on Extreme-scale Experiment-in-the-Loop Computing (XLOOP) | 978-1-6654-1122-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/XLOOP54565.2021.00008

Bridging Data Center AI Systems with Edge Computing for Actionable Information Retrieval

Zhengchun Liu*¶, Ahsan Ali*, Peter Kenesei†, Antonino Miceli†, Hemant Sharma†,
Nicholas Schwarz†, Dennis Trujillo†, Hyunseung Yoo*, Ryan Coffee‡, Naoufal Layad§,
Jana Thayer‡, Ryan Herbst‡, Chunhong Yoon‡, Ian Foster*¶
*Data Science and Learning Division, Argonne National Laboratory, Lemont, IL 60439, USA
†X-ray Science Division, Argonne National Laboratory, Lemont, IL 60439, USA
‡SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA
§Stanford University, Stanford, CA 94305, USA
¶University of Chicago, Chicago, IL 60637, USA

***Abstract*—Extremely high data rates at modern synchrotron and X-ray free-electron laser light source beamlines motivate the use of machine learning methods for data reduction, feature detection, and other purposes. Regardless of the application, the basic concept is the same: data collected in early stages of an experiment, data from past similar experiments, and/or data simulated for the upcoming experiment are used to train machine learning models that, in effect, learn specific characteristics of those data; these models are then used to process subsequent data more efficiently than would general-purpose models that lack knowledge of the specific dataset or data class. Thus, a key challenge is to be able to train models with sufficient rapidity that they can be deployed and used within useful timescales. We describe here how specialized data center AI (DCAI) systems can be used for this purpose through a geographically distributed workflow. Experiments show that although there are data movement cost and service overhead to use remote DCAI systems for DNN training, the turnaround time is still less than 1/30 of using a locally deploy-able GPU.**

I. Introduction

The increased coherence and brilliance of next-generation X-ray light sources, such as the upgraded Advanced Photon Source and the Linac Coherent Light Source, APS-U and LCLS-II respectively, will lead to increasingly large and rich data sets produced at high data rates. Such multi-modal data, captured in situ, can provide new insights into rare events such as crack initiation and phase transformations. However, the complexity and velocity of these data means that extracting the desired physical information becomes a considerable computing challenge, particularly when information is needed rapidly, for example to steer experiments. This data extraction challenge is not easily addressed by using conventional analytical methods. Such methods take too long to run on individual processors, and even if efficiently parallelizable (factors like irregular memory accesses often make them latency bound), they still need many processors to complete analyses in near real-time [1]. Fortunately, machine learning (ML)-based surrogate models can provide an effective alternative to conventional methods in these contexts. A suitably trained ML surrogate can approximate the results of an analytical method with high accuracy, and while it may perform more floating-point operations that the analytical method, can be executed at high speeds on specialized AI accelerators such as GPU, TPU, and NPU [2], which are sufficiently inexpensive to deploy near to data sources.

Corresponding Email: zhengchung.liu@anl.gov

There is a growing body of work on AI/ML methods for processing of X-ray source data for tomography [3], [4], [5], serial crystallography [6], [7], X-ray diffraction [8], [9], and source diagnostics themselves [10], [11]. For example, Pelt et al. [3] and Liu et al. [4], [5] used deep convolutional neural networks to improve tomographic reconstruction from limited measurements (e.g., sparse projections, short exposure time, or limited angle). Others have used DNNs to guide data collection under budgeted dosage [12] and to enhance streaming tomography [4], [5]. Using a pre-trained DNN model as a prior, Aslan et al. incorporated learned priors into the generic reconstruction framework for the joint ptycho-tomography problem [13]. Abeykoon el al. have deployed models to AI accelerators (such as edge TPU and Jetson) for low-cost data processing at the edge [14]. These methods, when combined with AI accelerators, thus make it feasible, in principle, to analyze data rapidly, at or near the point of data acquisition, rather than streaming them to a remote HPC system.

In order to use ML models effectively, we also need to be concerned with *retraining*, for example when a sample or experiment setup changes. *Training* involves the use of one of a variety of techniques to learn, from supplied training data (often a collection of feature–target pairs), a mapping function from features to targets. The output of this process is a trained model that can then be applied to other features (a process often referred to as *inference*) to obtain an estimation of the corresponding targets. In the case of deep neural networks (DNNs), training uses a computationally intensive process called stochastic gradient descent to solve the optimization problem of finding good DNN parameter values.

II. Motivations, Solution, and Gaps

We define a data center AI (DCAI) system as an AI accelerator that must be deployed in a data center due to its cooling, power supply, ventilation, and fire suppression requirements. DCAI systems can train ML models much more

978-1-6654-1122-6/21/$31.00 ©2021 IEEE
DOI 10.1109/XLOOP54565.2021.00008

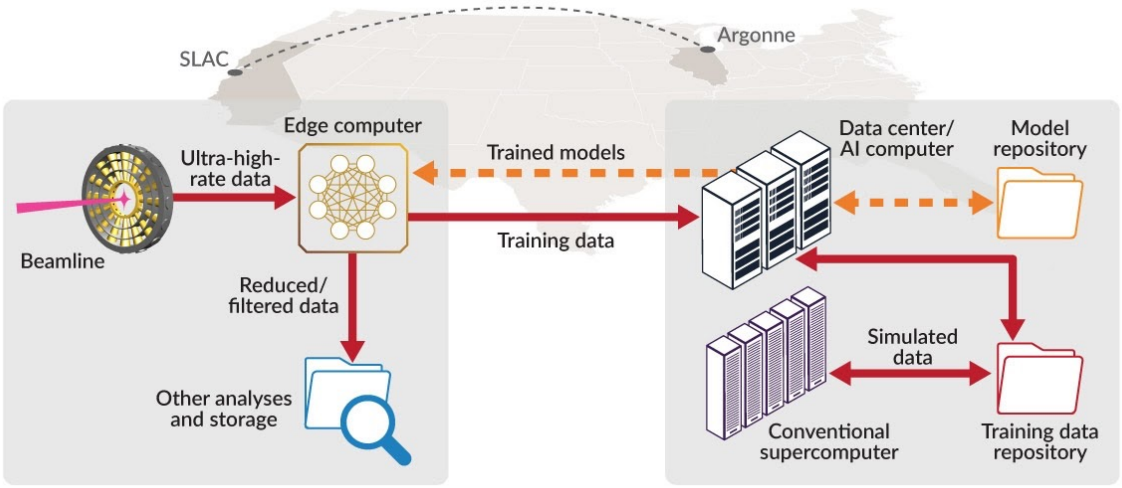

Fig. 1: A demonstration of the actionable information architecture uses an edge computer co-located with the experimental apparatus (CookieBox indicated here, the SLAC LCLS-II TMO beamline) for rapid reduction/filtering of high-velocity data; the ML model used to perform reduction/filtering is trained on a remote data center AI computer (here, at the Argonne Leadership Computing Facility). Solid crimson arrows are data flows; dashed orange are models.

rapidly than computing clusters that are maximally deployable within the experiment facility. Once the DNN is trained, we use another set of AI accelerators specialized for model inference, called `edge-AI` (e.g., FPGA, GPU with Tensor cores, edge TPU [14]), to process experiment data near the data acquisition in real-time. Since model inference is much less computing intensive than model training and it only needs to be as fast as the data generation rate (i.e., real/wall- time speed), `edge-AI` can be lightweight enough to be deployed within the experiment facility. Thus, in order to bridge the divide of powerful AI systems deployed in data center (e.g., scientific computing facility) and rapid ML model (re)training needs at experimental facility, we need to have a workflow, as shown in Figure 1, that is geographically distributed on multiple science facilities to (re)train ML models rapidly.

The basic concept of utilizing ML for real-time actionable information retrieval is the same irrespective of the application, as illustrated in Figure 1: data generated by a simulation close to the experiment, and/or collected in early stages of an experiment, and/or data from past similar experiments, are used to train models that, in effect, learn specific characteristics of the supplied training data; these models can then be used to process subsequent data more efficiently than general-purpose models that lack knowledge of the specific datasets or classes of data [5], [2], [15]. For example, in High-Energy X-Ray Diffraction Microscopy (HEDM), a single scan of 1440–3600 frames can be obtained in 5–10 minutes now and in perhaps 50–100 seconds at APS-U. When measuring a single sample on a layer-by-layer basis, similar data quality is observed repeatedly. Thus, an AI model trained on early layers can be used to process latter layers. In serial crystallography [16], [17], Bragg spots from past experiments can be used for training detectors for use in the current experiment. In single particle imaging [18], [19], the particles are unique for each experiment; realistic simulations can be used to train a neural network for classifying images of interest. In ptychography, the diffraction patterns collected at early stages of an experiment and their corresponding phase retrieved using the conventional solution of an inverse problem can be used to train a ML model, such as PtychoNN [15], and the trained model can then be used for more rapid phase retrieval from subsequent diffraction patterns.

Such applications in which a DNN is (re)trained for actionable information retrieval at the edge, near to a data sources, usually has time constraints that require us to deliver the model within a limited time-frame, particularly in the case of in-situ experiments. A key challenge is thus to be able to train models quickly enough that they can be deployed and used within useful timescales. The emergence of purpose-built accelerators (e.g., TPU [20], Cerebras[21], SambaNova[22]) for high-speed, high-velocity, and/or big data AI models allows for scenarios in which model training is performed on a specialized accelerator while inference is performed at the edge, near the data source, for real-time data analysis. However, usually it is not feasible to deploy these AI systems near an experiment facility because AI systems usually require a significant amount of hardware and software infrastructure, including power subsystems, stable and uninterruptible power supplies, proper ventilation, high-quality cooling systems, fire suppression, reliable backup generators, and connections to external networks. Instead these AI systems are usually deployed in a data center, for which reason we refer to them as data center AI (DCAI) systems. Moreover, there is also a strong economical argument of using DCAI systems, i.e. allowing to share the very expensive specialized AI processors between experiments in multiple facilities.

In this paper, we propose an approach to the use of DCAI systems can be integrated into geographically distributed **workflow** solutions to facilitate model training on remote DCAI systems and model deployment on edge devices. Thus, one of the important elements to retrieve actionable information in real time is the use of powerful DCAI systems to enable rapid (re)training of ML/AI models that are then deployed on edge devices for production use, as shown in Figure 1. A second major driver for the use of data center computers is for the generation of simulated data for model training. Here we face two distinct needs: high-throughput runs to generate large quantities of simulation data, and low-latency runs to generate results rapidly during an experiment.

III. Workflow Design and Implementation

We used the Globus Flows [23], funcX [24], and Globus file transfer [25] services to build a workflow to automatically (re)train DNNs with given data or simulations using DCAI and HPC. In this section, we briefly introduce key features and functionalities of each service and describe how the geographically distributed workflow is realized using these services.

The Globus **Flows** service introduces *Action Providers*, *Actions*, and *Flows* to create custom processes solving particular research data management problems [23], [26]. An *Action Provider* is an HTTP accessible service which acts as a single step in a process and implements the Action Provider Interface. An *Action* represents a single, discrete invocation of an Action Provider. A *Flow* represents a single process that orchestrates a series of services/actions into a self contained operation. One can think of a Flow as a declaratively defined ordering of Action Providers with condition handling to define expected success or failure scenarios. Globus Auth [27] is used to authenticate all interactions with Action Providers, Actions and Flows.

funcX provides the function-as-a-service capability to execute functions across a federated ecosystem of funcX endpoints [24]. It basically offers the ability to turn any computing resource, including clouds, clusters, supercomputers, edgeAI devices and DCAI systems into a function-serving endpoint by deploying a Python-based funcX endpoint to the target system. So we can then build our computation actions, including simulation, data annotation and model training, using funcX service and wrap it as an Action of Globus Flows. The Flow engine will then interact with funcX service to automatically execute the function (action). More importantly, funcX is serverless and provides the fire-and-forget convenience for user.

The **Globus** transfer service provides a secure, unified interface to the data. It allows us to start and manage transfers between endpoints, while automatically tuning parameters to maximize bandwidth usage, managing security configurations, providing fault recovery, and notifying users of completion and problems. We use Globus to transfer data and trained DNNs between systems within and across organizations by wrapping each file transfer request as an action of Globu Flows.

Figure 2 shows the overall architecture of the system. Basically, all computing functions, e.g., simulation for training data generation, data curation and model training, are abstracted as a funcX function and wrapped as an action of Globus Flows; and all data dependencies are solved by wrapping the data transfer as an action of Globus Flows using Globus file transfer service. A developer builds and deploys the workflows to the **Flows** service and shares it with actual users (e.g., experimental scientists). The user only needs to interact with the workflows through a client to initiate the training task and the flow engines will orchestrate resources with action providers to run the workflow and deliver the trained model to the place that is defined in the workflow by user.

The implementation of the workflow shown in this paper is available at https://github.com/AISDC/DNNTrainerFlow and one can use it as a reference to rebuild their own workflow if similar building blocks are used. A recording of the demonstration of the workflow that bridges SLAC experimental facility and Argonne Leadership Computing Facility(ALCF) is available at https://youtu.be/lL6HsIk3xjE.

IV. Analytical Modeling and Evaluation

A. Performance Modeling

We construct the various applications that we consider here in terms of the following six basic operations:

i) **C**ollect a datum, d or
ii) **S**imulate an experiment to generate a datum, d, without an experiment;
iii) **A**nalyze a datum using a conventional algorithm (e.g., Bragg peak localizing using pseudo-Voigt), generating an analysis (e.g., Bragg peak locations), a;
iv) **T**rain (or retrain) a ML model with some number of $\{d, a\}$ pairs, generating a model, m;
v) **D**eploy the ML model, m, to an edge-AI device; and
vi) **E**stimate an analysis with a previously trained ML model, generating an estimated analysis, $\hat{a}$.

When an operation may be performed at different locations, we denote that by a subscript, such as $\mathbf{A}_{\text{ex}}$ for an analysis performed on a computer at an experiment and $\mathbf{A}_{\text{dc}}$ for an analysis performed in a data center. We represent the movement of data d from location a to location b as $a \xrightarrow{d} b$. Finally, we define $\mathcal{C}(o)$ to be the cost of an operation o. Thus, for example, the conventional way of moving data from experiment to data center for analysis, and then returning result to the experiment (e.g., for steering), will cost, as a function of d

$$f^c(d) = \mathcal{C}(ex \xrightarrow{d} dc) + \mathcal{C}\left(\mathbf{A}_{\text{dc}}(d)\right) + \mathcal{C}(dc \xrightarrow{a} ex), \qquad (1)$$

while performing analysis at the experiment facility will cost

$$f(d) = \mathcal{C}(\mathbf{A}_{\text{ex}}), \qquad (2)$$

and estimating an analysis at the experiment will cost $\mathcal{C}(\mathbf{E}_{\text{ex}})$.

In comparison for processing the same amount of data d, we first move a data subset, $\bar{d}$, from experiment to data center for analysis and train a model, and then returning a model for **E**stimating the subsequent analysis will cost, as a function of d:

$$f^{ml}(d) = \mathcal{C}(ex \xrightarrow{\bar{d}} dc)) + \mathcal{C}(\mathbf{A}_{\text{dc}}(\bar{d})) + \mathcal{C}(\mathbf{T}_{\text{da}}(\bar{d})) + \mathcal{C}(dc \xrightarrow{m} ex) + \mathcal{C}(\mathbf{E}_{d-\bar{d}}) \qquad (3)$$

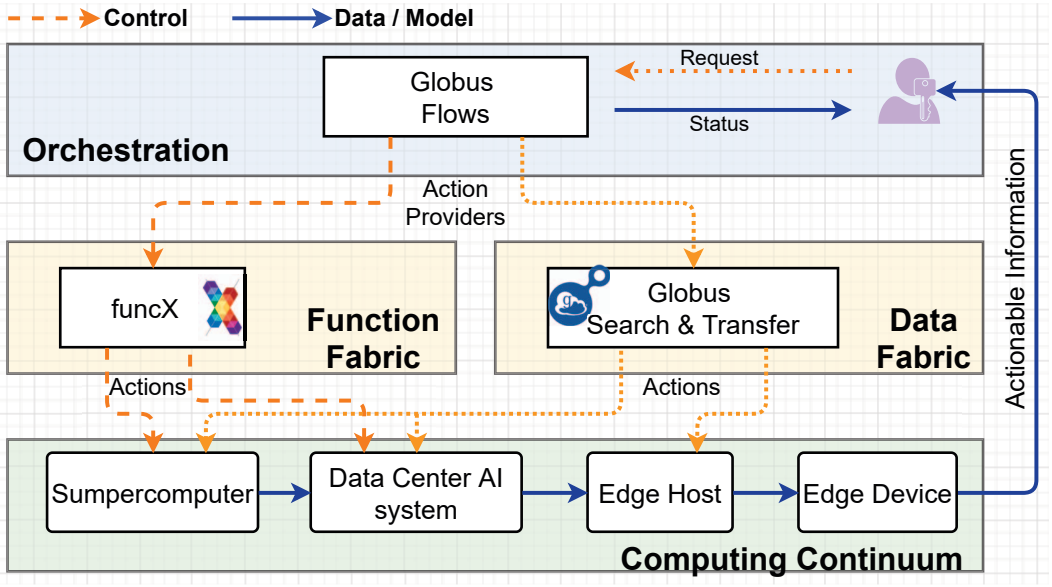

Fig. 2: Architecture and building blocks of the workflow. Solid arrows are data flows; dashed arrows are control flows.

The costs for **C**, **S**, **A**, and **E** are deterministic when dedicated resources are used and thus only need to be profiled once for a given type of experiment. The cost of **T** depends on the specific AI system used, it can range from seconds to hours, as we will discuss in §V. Wide-area network congestion [28], [29] can result in variability in the data movement cost, $\mathcal{C}(a \xrightarrow{d} b)$, but most research and education networks (e.g., ESnet and Internet2) are over-provisioned in order to support capacity bursts [30], [31], [32]. For example, when the backbone circuit of Internet2 is seen regularly to sustain 40% utilization, staff initiate a backbone augment [31]. Moreover, previous work shows that the model for $a \xrightarrow{t} b$ can be approximated by using a linear model $T = \frac{1}{v}x + S$ where x is the bytes in total, v is average transfer rate and S is a constant start up cost mainly depends on the number of files in the dataset [33], [34]. More complicated machine learning based prediction model can also be incorporated for more congested network and data/or data transfer nodes [35].

B. Model based Analysis

The analytical model proposed in §IV-A can be used to evaluate different data analysis solutions (e.g., Equations 1–3) for a given experiment. As the cost of **T**, we will demonstrate the evaluation by using HEDM as an example in §V.

Here we use BraggNN [2], an ML-based surrogate solution for Bragg peak analysis (see §V-B), as an example to evaluate the most proper way of data analysis. As reported by Liu et al. [2], **A** takes about 2000 core×seconds for 800 000 peaks, while **E** with BraggNN can process 800 000 peaks in 280 ms (batch processing). Here we assume that there is a CPU cluster with 1024 usable cores (to our knowledge, GPU accelerated **A** does not exist, mostly because its computing and memory access pattern do not fit well with GPU architecture). Thus, we get $\mathcal{C}\left(\mathbf{A}_{\text{dc}}(d)\right) = 2.44\mu s$, and $\mathcal{C}\left(\mathbf{E}_{\text{dc}}(d)\right) = 0.35\mu s$ where d denotes one patch of Bragg peak with 11 by 11 pixels, 16 bits per pixel.

The backbone fiber connects experimental facility (e.g., SLAC here) and data center (e.g., ALCF here) is 100 Gbps. We benchmarked the file transfer throughput using Globus with one DTN with a 10 Gbps network card at each side and presented results in Figure 3 with different level of parallelism. As one can see, we can get more than 1GB/s when transfer

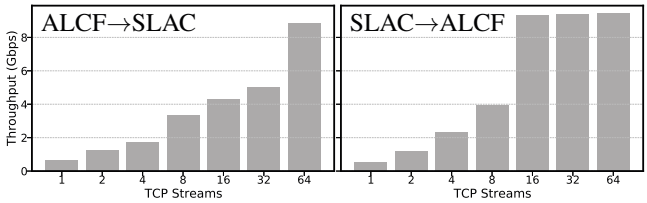

Fig. 3: File Transfer performance between ALCF and SLAC.

multiple files concurrently (detailed study in [36]). As discussed in §IV-A, the network is over-provisioned, we conservatively assume that we can get 1 GB/s at most time. Thus we have $\mathcal{C}(ex \xrightarrow{d} dc)) = \mathcal{C}(dc \xrightarrow{d} ex)) = 0.24\mu s$. As modeled in Equation 3, we need to transfer a portion, p, of our raw data (say p=10%) to data center, label it using **A** and then train BraggNN with **T**. Both the trained BraggNN model (3 MB, i.e., 3000 μs) and the labeled portion (each datum leads to 8 bytes) need to be transferred back to SLAC. As will be discussed in Table I, BraggNN can be trained in as little as 19 seconds when using the Cerebras system. We thus assume $\mathcal{C}(\mathbf{T}_{\text{da}}(\bar{d})) = 19s$. If the dataset to be processed has N datum, then the total time using conventional way with a cluster at data center will take

$$f^c(d) = N * 0.24 + N * 2.44 + N * 8 * 10^{-3} \quad (\mu s), \quad (4)$$

while the time when use BraggNN will be

$$f^{ml}(d) = p * N * 0.24 + p * N * 2.44 + p * N * 8 * 10^{-3} + 19 * 10^{6} + 3000 + (1-p) * N * 0.35 \quad (\mu s). \tag{5}$$

The number of Bragg peaks per experiment ranges from tens of hundred thousands to millions. We vary N and compare conventional solution with ML surrogate based solution in Figure 4. As there is a static cost to train the BraggNN, the conventional solution outperforms proposed solution only when the number of data is small. The analytical model can thus be used to decide which solution to take before processing.

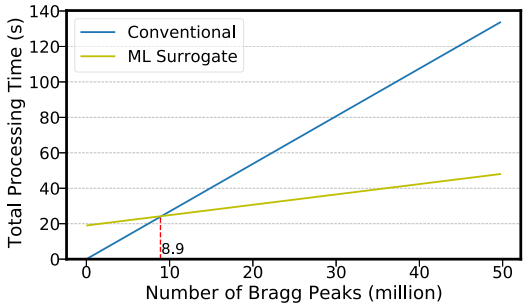

Fig. 4: A comparison of conventional vs. ML-based surrogate solution for processing datasets of varying size.

V. Experiment and Discussion

In this section, we will demonstrate the workflow designed for rapid (re)training of two DNNs with the above discussed steps. Here we care more about the end-to-end to (re)train a machine learning model with new dataset from a user's point of view. The end-to-end is defined as the time difference from a user initializing the model (re)training with new dataset till the trained model is received at edge host machine of the user's choice. We will compare the end-to-end time of the proposed solution using remote DCAI systems against an high-end GPU that can be deployed locally within the experiment facility.

A. Experiment Setup

This experiment involves a case in which SLAC needs to (re)train DNNs for an experiment. We compare and contrast two scenarios: one in which model training is performed remotely, using a Cerebras or SambaNova DCAI system at ALCF, and one in which training is performed locally, on an NVIDIA V100 GPU co-located with the experiment.

In the remote case, the DCAI system is 3000 km distant, with a network round trip time over the Energy Sciences Network (ESNet, a high-speed computer network serving United States Department of Energy scientists and their collaborators worldwide [32]) of about 48 ms at a peak bandwidth of 100 Gbps. We must transfer the training dataset produced at SLAC to ALCF, train the model with the dataset, and transfer the trained model back to SLAC. In the local case, there is no cost for wide-area data transfer.

B. Deep Neural Networks

In order to demonstrate the concept of using remote DCAI for rapid model (re)training, we used two DNNs, BraggNN [2] and CookieNetAE, as examples to demonstrate the superiority of using remote DCAI system for retraining. Both models, detailed as follows, are purposely designed to run inference at edge to retrieve actionable information from experimental data in real time, and both need retraining time to time.

a) CookieNetAE: The CookieBox detector [11] is an angular array of sixteen electron Time-of-Flight (eToF) spectrometers. The x-ray shot photo-ionizes gas molecules in the interaction point, ejecting electrons. These electrons drift through a series of electrostatic potential plates in the 16 channels and then detected by microchannel plates (MCP). This problem becomes difficult when we consider using a circularly polarized optical laser field in the interaction region and when the number of detected electrons is low. CookieNetAE is a deep neural network designed to estimates the energy-angle dependent probability density function of electrons' energy for all 16 channels. CookieNetAE takes an input image where each row in the image corresponds to an empirical energy histogram with 128 bins of 1 ev width of a given CookieBox channel built after the time-energy mapping. And the output will be an image containing the probability density of electrons' energy in each channel. This model has 8 convolution layers totals 343,937 trainable parameters, rectifier activation function was used for all layers. The mean squared error was used as the loss function and the Adam optimizer with a learning rate of 0.001 was used to train the model.

b) BraggNN: X-ray diffraction based microscopy techniques such as HEDM rely on knowledge of the position of diffraction peaks with high precision. These positions are typically computed by fitting the observed intensities in area detector data to a theoretical peak shape such as pseudo-Voigt. As experiments become more complex and detector technologies evolve, the computational cost of such peak detection and shape fitting becomes the biggest hurdle to the rapid analysis required for real-time feedback during in-situ experiments. BraggNN [2] is a deep learning model designed to localize Bragg peak positions much more rapidly than conventional pseudo-Voigt peak fitting. Recent advances in deep learning method implementations and special-purpose model inference accelerators allow BraggNN to deliver enormous performance improvements relative to the conventional method, running, for example, more than 200 times faster than a conventional method on a consumer-class GPU card with out-of-the-box software.

C. Results and discussion

We conducted our experiments on three DCAI systems: multi-GPU server, Cerebras [21] and SambaNova [22] all located at ALCF. Horovod [37] was used to implement DNN training using multi-GPU. The entire wafer of the Cerebras was used for data parallelism via model replica.

Table I presents the time breakdown of $\mathcal{C}(\mathbf{T})$ for each of the actions/steps in the workflow and compares it with training the

TABLE I: Time breakdown of the workflow steps/actions when using either a remote Cerebras DCAI system, a remote 8-GPU server or a remote SambaNova (only used 1 out of 8 RDUs per node) versus using one local GPU. The purpose of listing the performance of different systems is not to compare them, as this is not a systematical benchmarking study. The purpose is rather to demonstrate the feasibility of using powerful yet remote DCAI systems for AI at edge applications.

Mode \ Time	Neural Network	Data Transfer (s)	Model Training (s)	Model Transfer (s)	End-to-End (s)
Local (one GPU)	BraggNN [2]	N/A	1102	N/A	1102
Remote (Cerebras, Entire Wafer)		7	19	5	31
Remote (SambaNova, 1-RDU)		7	139	5	151
Local (one GPU)	CookieNetAE	N/A	517	N/A	517
Remote (Cerebras, Entire Wafer)		5	6	4	15
Remote (multi-GPU server)		5	88	4	97

same model with the same dataset using a single local NVIDIA V100 GPU that can be deployed locally within the experiment facility. As one can see, although the data and model transfer (i.e., cost when use remote resources) accounted nearly half of the end-to-end time in certain cases, using a remote DCAI system is still more than 30 times faster than using local resources for both BraggNN and CookieNetAE.

We note that only one (out of eight per node) Reconfigurable Dataflow Unit (RDU) is used for the BraggNN based experiment on SambaNova system. Experiment for data parallelism over multiple RDUs is forthcoming. We did not conduct experiments for BraggNN using multi-GPU because BraggNN is light-weight by design that will be latency-bounded when perform data parallelism over multiple(distributed) GPUs, i.e., the speedup of computing gaining from using multiple GPUS is less than the necessary cost on gradients synchronization (`allreduce` operation). We leave further optimization of BraggNN trained on multiple distributed GPUs for future work.

VI. Related work

Real-time scientific experiment data analysis leveraging data center resource such as cloud or supercomputer has been an active area of research to fulfill the increasingly data-intensive analysis demanding from upgrades to science infrastructure. Wilamowski et al. [38] used Globus Flows to automate real-time SSX data analysis, but did not use ML models or AI accelerators as here. Rocki et al. [39] discussed the possibility of using DCAI systems for PDE codes in scientific applications, and demonstrated the benefit over using conventional CPU or GPU based solutions. Emani et al. [40] explored the suitability of SambaNova, another DCAI system, for diverse AI for Science workloads and observed significant performance gains over traditional hardware. Acciarri et al. [41] advanced state-of-the-art accuracy for an important neutrino physics image segmentation problem leveraging the large memory of DCAI systems which are not possible to fit on the highest-end GPU because of the large tensor size (convolutions neural networks with images beyond 50k x 50k resolution).

Thorpe et al. [42] also used function-as-a-service computing (AWS Lambda) offered by cloud computing provider to scale and accelerate machine learning modeling training task. In the context of end-to-end distributed science workflows, Salim et al. [43] proposed the Balsam workflow framework to enable wide-area, multi-tenant, distributed execution of analysis jobs on DOE supercomputers.

VII. Conclusion and Future work

We have presented an automated workflow for rapid deep neural networks training using remote resources, which automates the model (re)training and achieves a turnaround time between initiating and model delivery to edge host of less than 151 seconds including all data movement overhead. As a comparison, it takes $\sim$17 minutes (though no cost on data movement) when training the same model using one high-end GPU that is deploy-able within the data acquisition machine. This workflow proves the feasibility of using powerful yet remote data center AI systems to enable rapid (re)training of deep neural networks for production use on edge devices. We automated the rapid DNN training, using remote data center AI (DCAI) systems (e.g., Cerebras and SambaNova). The workflow also simplifies the use of remotely hosted DCAI systems for domain scientists.

As for future work, there are three directions that we are currently pursuing. 1), we are building the model repository, as shown in Figure 1, so as to pick up the right model as foundation to fine-tune using new dataset instead of retraining from scratch, to further accelerate the training process. 2), we are also building a data repository to augment training dataset or substitute unlabelled dataset, because the labelling process, $\mathcal{C}(\mathbf{A}_{\text{dc}}(\bar{d}))$, is usually time consuming (which motivates ML-based surrogate solution). Both 1) and 2) will need data center (HPC, DCAI and cloud) resource. 3), in most cases the new experimental data to (re)train the DNN are not labelled, we will need to run labelling algorithm $\mathbf{A}$ (e.g., pseudo-Voigt profiling for BraggNN [2]). As the training process is mini-batch based which can be started before getting all training samples, we can try to partially overlap $\mathbf{A}$ and $\mathbf{T}$ in the workflow to shorten end-to-end time.

APPENDIX: REPRODUCIBILITY

A. Dependencies

The workflow runs in the Globus[1] toolkit ecosystem. A Globus account is needed so as to run the workflow to reproduce results reported in this paper. One can either create one using any email address as ID, or through existing organizational login if participated. Both are free of charge.

a) funcX [24]: is a distributed Function as a Service (FaaS) platform that enables flexible, scalable, and high performance remote function execution[2]. It is one of the action providers of the workflow developed in this paper to provision computing resources. We installed FuncX server funcX-endpoint version 0.3.2 at all data center systems used in this paper, e.g., via `pip install funcx_endpoint`. Then we deploy a funcX endpoint by running `funcx-endpoint configure endpoint-name` and following the authentication instruction. A UUID will be generated if succeeded, then the endpoint UUID is the unique ID that will be supplied as an argument to the workflow to provision computing resources associated with the endpoint to train the DNN.

Similarly, we need to install FuncX client (i.e., using `pip install funcx`) of the same release version on the machine that the user will initialize and start running the workflow. The client version is needed to register functions to FuncX service so as to supply the function ID to the workflow.

b) Globus Automate: The Client SDK of globus Automate provides Python interface for working with Globus Automate, primarily Globus Flows[3]. We need to install the SDK on the machine that will be used to built and/or run the workflow via `pip install globus-automate-client`.

B. Run the Proposed Workflow

The implementation of the proposed workflow is open source at https://github.com/AISDC/DNNTrainerFlow as a Jupyter notebook. `DNN-Train-Flow.ipynb`. There are three major components to build the workflow detailed as follows.

- Flow definition which basically defines the order and data dependencies (i.e., arguments of each action) of each action. This can be defined using a Python dictionary and the sample can be found from the open source repository.
- A self-contained (all modules should be imported within the function) Python function registered to FuncX service to train the model.
- Arguments of the workflow which basically contains the full path to the train data and the full path to receive the trained DNN. All arguments are organized in a Python dictionary.

Once the workflow is built and registered to the Globus Flow service, only the returned ID is needed to run it as many times as needed with augments supplied, i.e., similar as running a function with different arguments. So we build the workflow once and then share it with user to run it with their own arguments to (re)train their model with supplied data and receive the trained model at the specified destination.

C. Reproduce local GPU results

The BraggNN [2] model as well as the training dataset that we used to demonstrate the workflow is publicly available at https://github.com/lzhengchun/BraggNN. One needs to install pyTorch=1.9.0, h5py=2.10.0 and numpy=1.19.2 to reproduce the local single GPU results reported in Table I by running the `main.py` or `horovodrun -np 8 python main-hvd.py` with default arguments.

Similarly, the code and a sample dataset of CookieNetAE can be downloaded from https://github.com/AISDC/CookieNetAE and run `horovodrun -np 1 python main-hvd.py` locally to reproduce training using local resource.

D. Reproduce remote GPU cluster results

As GPU cluster is the most accessible resource among all three AI systems we used in this paper, here we will detail the steps to reproduce the results reported in this paper about using GPU cluster. The funcX functions are executed under the same Python environment as the funcx-endpoint installation with default configuration. As DNN training, especially the data parallelism or AI system based training, usually needs special Python environment. We invoke the training script via system call wrapped in the funcX function.

We used Horovod [37] (v0.22.1, installation guide[4]) to make use of multiple GPUs via data parallelism. One can then run the workflow by supplying `horovodrun -np 8 python main-hvd.py` as the funcX function argument for the system call, together with other workflow augments to reproduce results reported in Table I. Specific instructions for BraggNN and CookieNetAE are detailed as follows:

1) BraggNN: The repository at https://github.com/lzhengchun/BraggNN hosts the code and dataset, `main-hvd.py`, for training BraggNN using data parallelism via Horovod. The distributed data parallel version of BraggNN using Horovod can support multiple GPUs on multiple nodes. As we discussed in §V-C, one will not see much acceleration when training BraggNN using multiple GPUs. We leave further optimization effort for future work.

2) CookieNetAE: The Horovod based implementation of CookieNetAE and sample dataset are available at https://github.com/AISDC/CookieNetAE. Similar as it for BraggNN, one can reproduce results in §V-C by supplying `horovodrun -np 8 python main-hvd.py` to the workflow argument dictionary.

E. SambaNova and Cerebras

Non-disclosure agreements with SambaNova and Cerebras prevent us from sharing our code for these two systems at this moment. As these AI system also have their own Python environment, the way of integration these AI system with the workflow is exactly the same as it for multi-GPU (i.e., via a system call). So, we assume one can reproduce similar results

[1] https://www.globus.org
[2] https://funcx.readthedocs.io
[3] https://globus-automate-client.readthedocs.io
[4] https://horovod.readthedocs.io

as we reported using any other DNNs if SambNova and/or Cerebras resource is available.

ACKNOWLEDGEMENTS

This material was based upon work supported by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357 and Office of Basic Energy Sciences under Award Number FWP-35896. This research used resources of the Argonne Leadership Computing Facility, a DOE Office of Science User Facility supported under Contract DE-AC02-06CH11357. The demonstration models were developed with support also for the specific detector development for attosecond methods at LCLS-II under Office of Basic Energy Sciences under Award Number FWP-100498. Much of the computational resources at SLAC were funded under an LDRD award for EdgeML. CookieNetAE development was funded under Office of Basic Energy Science award FWP-100498 and FWP-100643. The authors thank Ryan Chard (Argonne National Laboratory) for assistance with funcX. We thank the engineering support from the Cerebras Systems Inc. and SambaNova Systems. We thank three anonymous reviewers for their constructive comments.

REFERENCES

[1] T. Bicer, D. Gürsoy, V. De Andrade, R. Kettimuthu, W. Scullin, F. De Carlo, and I. T. Foster, "Trace: A high-throughput tomographic reconstruction engine for large-scale datasets," *Advanced Structural and Chemical Imaging*, vol. 3, no. 1, pp. 1–10, 2017.

[2] Z. Liu, H. Sharma, J.-S. Park, P. Kenesei, J. Almer, R. Kettimuthu, and I. Foster, "BraggNN: Fast x-ray Bragg peak analysis using deep learning," *arXiv preprint arXiv:2008.08198*, 2020.

[3] D. M. Pelt, K. J. Batenburg, and J. A. Sethian, "Improving tomographic reconstruction from limited data using mixed-scale dense convolutional neural networks," *Journal of Imaging*, vol. 4, no. 11, 2018. [Online]. Available: http://www.mdpi.com/2313-433X/4/11/128

[4] Z. Liu, T. Bicer, R. Kettimuthu, D. Gursoy, F. De Carlo, and I. Foster, "TomoGAN: Low-dose synchrotron x-ray tomography with generative adversarial networks," *Journal of the Optical Society of America. A, Optics, Image Science, and Vision*, vol. 37, no. 3, pp. 422–434, 2020.

[5] Z. Liu, T. Bicer, R. Kettimuthu, and I. Foster, "Deep learning accelerated light source experiments," in *IEEE/ACM Third Workshop on Deep Learning on Supercomputers*. IEEE, 2019, pp. 20–28.

[6] T.-W. Ke, A. S. Brewster, S. X. Yu, D. Ushizima, C. Yang, and N. K. Sauter, "A convolutional neural network-based screening tool for X-ray serial crystallography," *Journal of Synchrotron Radiation*, vol. 25, no. 3, pp. 655–670, 2018.

[7] A. Souza, L. B. Oliveira, S. Hollatz, M. Feldman, K. Olukotun, J. M. Holton, A. E. Cohen, and L. Nardi, "DeepFreak: Learning crystallography diffraction patterns with automated machine learning," *arXiv preprint arXiv:1904.11834*, 2019.

[8] F. Oviedo, Z. Ren, S. Sun, C. Settens, Z. Liu, N. T. P. Hartono, S. Ramasamy, B. L. DeCost, S. I. Tian, G. Romano *et al.*, "Fast and interpretable classification of small x-ray diffraction datasets using data augmentation and deep neural networks," *npj Computational Materials*, vol. 5, no. 1, pp. 1–9, 2019.

[9] P. M. Vecsei, K. Choo, J. Chang, and T. Neupert, "Neural network based classification of crystal symmetries from x-ray diffraction patterns," *Physical Review B*, vol. 99, no. 24, p. 245120, 2019.

[10] A. Sanchez-Gonzalez, P. Micaelli, C. Olivier, T. Barillot, M. Ilchen, A. Lutman, A. Marinelli, T. Maxwell, A. Achner, M. Agåker *et al.*, "Accurate prediction of X-ray pulse properties from a free-electron laser using machine learning," *Nature Communications*, vol. 8, no. 1, pp. 1–9, 2017. [Online]. Available: http://dx.doi.org/10.1038/ncomms15461

[11] A. Corbeil Therrien, R. Herbst, O. Quijano, A. Gatton, and R. Coffee, "Machine learning at the edge for ultra high rate detectors," in *IEEE Nuclear Science Symposium and Medical Imaging Conference*, 10 2019, pp. 1–4.

[12] Z. Wu, T. Bicer, Z. Liu, V. De Andrade, Y. Zhu, and I. T. Foster, "Deep learning-based low-dose tomography reconstruction with hybrid-dose measurements," in *IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments (MLHPC) and Workshop on Artificial Intelligence and Machine Learning for Scientific Applications (AI4S)*, 2020, pp. 88–95.

[13] S. Aslan, Z. Liu, V. Nikitin, T. Bicer, S. Leyffer, and D. Gürsoy, "Joint ptycho-tomography with deep generative priors," *Machine Learning: Science and Technology*, vol. 2, no. 4, p. 045017, 2021.

[14] V. Abeykoon, Z. Liu, R. Kettimuthu, G. Fox, and I. Foster, "Scientific image restoration anywhere," in *IEEE/ACM 1st Annual Workshop on Large-scale Experiment-in-the-Loop Computing*. IEEE, 2019, pp. 8–13.

[15] M. J. Cherukara, T. Zhou, Y. Nashed, P. Enfedaque, A. Hexemer, R. J. Harder, and M. V. Holt, "AI-enabled high-resolution scanning coherent diffraction imaging," *Applied Physics Letters*, vol. 117, no. 4, p. 044103, 2020.

[16] K. Nass, L. Redecke, M. Perbandt, O. Yefanov, M. Klinge, R. Koopmann, F. Stellato, A. Gabdulkhakov, R. Schönherr, D. Rehders, J. M. Lahey-Rudolph, A. Aquila, A. Barty, S. Basu, R. B. Doak, R. Duden, M. Frank, R. Fromme, S. Kassemeyer, G. Katona, R. Kirian, H. Liu, I. Majoul, J. M. Martin-Garcia, M. Messerschmidt, R. L. Shoeman, U. Weierstall, S. Westenhoff, T. A. White, G. J. Williams, C. H. Yoon, N. Zatsepin, P. Fromme, M. Duszenko, H. N. Chapman, and C. Betzel, "In cellulo crystallization of Trypanosoma brucei IMP dehydrogenase enables the identification of genuine co-factors," *Nature Communications*, vol. 11, no. 1, pp. 1–13, dec 2020.

[17] A. Meents, M. O. Wiedorn, V. Srajer, R. Henning, I. Sarrou, J. Bergtholdt, M. Barthelmess, P. Y. Reinke, D. Dierksmeyer, A. Tolstikova, S. Schaible, M. Messerschmidt, C. M. Ogata, D. J. Kissick, M. H. Taft, D. J. Manstein, J. Lieske, D. Oberthuer, R. F. Fischetti, and H. N. Chapman, "Pink-beam serial crystallography," *Nature Communications*, vol. 8, no. 1, p. 1281, dec 2017. [Online]. Available: http://www.ncbi.nlm.nih.gov/pubmed/29097720http://www.pubmedcentral.nih.gov/articlerender.fcgi?artid=PMC5668288

[18] M. M. Seibert, T. Ekeberg, F. R. Maia, M. Svenda, J. Andreasson, O. Jönsson, D. Odić, B. Iwan, A. Rocker, D. Westphal, M. Hantke, D. P. Deponte, A. Barty, J. Schulz, L. Gumprecht, N. Coppola, A. Aquila, M. Liang, T. A. White, A. Martin, C. Caleman, S. Stern, C. Abergel, V. Seltzer, J. M. Claverie, C. Bostedt, J. D. Bozek, S. Boutet, A. A. Miahnahri, M. Messerschmidt, J. Krzywinski, G. Williams, K. O. Hodgson, M. J. Bogan, C. Y. Hampton, R. G. Sierra, D. Starodub, I. Andersson, S. Bajt, M. Barthelmess, J. C. Spence, P. Fromme, U. Weierstall, R. Kirian, M. Hunter, R. B. Doak, S. Marchesini, S. P. Hau-Riege, M. Frank, R. L. Shoeman, L. Lomb, S. W. Epp, R. Hartmann, D. Rolles, A. Rudenko, C. Schmidt, L. Foucar, N. Kimmel, P. Holl, B. Rudek, B. Erk, A. Hömke, C. Reich, D. Pietschner, G. Weidenspointner, L. Strüder, G. Hauser, H. Gorke, J. Ullrich, I. Schlichting, S. Herrmann, G. Schaller, F. Schopper, H. Soltau, K. U. Kühnel, R. Andritschke, C. D. Schröter, F. Krasniqi, M. Bott, S. Schorb, D. Rupp, M. Adolph, T. Gorkhover, H. Hirsemann, G. Potdevin, H. Graafsma, B. Nilsson, H. N. Chapman, and J. Hajdu, "Single mimivirus particles intercepted and imaged with an x-ray laser," *Nature*, vol. 470, no. 7332, pp. 78–82, feb 2011.

[19] A. Hosseinizadeh, G. Mashayekhi, J. Copperman, P. Schwander, A. Dashti, R. Sepehr, R. Fung, M. Schmidt, C. Yoon, B. Hogue, G. Williams, A. Aquila, and A. Ourmazd, "Conformational landscape of a virus by single-particle x-ray scattering," *Nature Methods*, vol. 14, no. 9, 2017.

[20] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P.-l. Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, D. Killebrew, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, J. Ross, M. Ross, A. Salek, E. Samadiani, C. Severn, G. Sizikov, M. Snelham, J. Souter, D. Steinberg, A. Swing, M. Tan, G. Thorson, B. Tian, H. Toma, E. Tuttle, V. Vasudevan, R. Walter, W. Wang, E. Wilcox, and D. H. Yoon, "In-datacenter performance analysis of a Tensor Processing Unit," *SIGARCH Comput. Archit. News*, vol. 45, no. 2, p. 1–12, Jun. 2017. [Online]. Available: https://doi.org/10.1145/3140659.3080246

[21] Cerebras, "Cerebras Systems," https://cerebras.net, accessed: 2021-08-18.
[22] SambaNova, "SambaNova Systems," https://sambanova.ai, accessed: 2021-08-18.
[23] R. Chard, K. Chard, and I. Foster, "Globus Automate: A distributed research automation platform," *figshare*, 2019.
[24] R. Chard, Y. Babuji, Z. Li, T. Skluzacek, A. Woodard, B. Blaiszik, I. Foster, and K. Chard, "FuncX: A federated function serving fabric for science," in *Proceedings of the 29th International Symposium on High-Performance Parallel and Distributed Computing*, 2020, pp. 65–76.
[25] I. Foster, "Globus Online: Accelerating and democratizing science through cloud-based services," *IEEE Internet Computing*, vol. 15, no. 3, pp. 70–73, 2011.
[26] R. Ananthakrishnan, B. Blaiszik, K. Chard, R. Chard, B. McCollam, J. Pruyne, S. Rosen, S. Tuecke, and I. Foster, "Globus platform services for data publication," in *Practice and Experience on Advanced Research Computing*, ser. PEARC '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3219104.3219127
[27] S. Tuecke, R. Ananthakrishnan, K. Chard, M. Lidman, B. McCollam, S. Rosen, and I. Foster, "Globus Auth: A research identity and access management platform," in *IEEE 12th International Conference on e-Science (e-Science)*. IEEE, 2016, pp. 203–212.
[28] Z. Liu, P. Balaprakash, R. Kettimuthu, and I. Foster, "Explaining wide area data transfer performance," in *26th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '17. New York, NY, USA: ACM, 2017, pp. 167–178. [Online]. Available: http://doi.acm.org/10.1145/3078597.3078605
[29] Z. Liu, R. Kettimuthu, I. Foster, and N. S. V. Rao, "Cross-geography scientific data transferring trends and behavior," in *27th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '18. New York, NY, USA: ACM, 2018, pp. 267–278. [Online]. Available: http://doi.acm.org/10.1145/3208040.3208053
[30] Y. Huang and R. Guerin, "Does over-provisioning become more or less efficient as networks grow larger?" in *13TH IEEE International Conference on Network Protocols*. IEEE, 2005, pp. 11–pp.
[31] Internet2, "Internet2 IP Backbone Capacity Augment Practice," https://internet2.edu/community/about-us/policies/internet2-ip-backbone-capacity-augment-practice/, accessed: 2021-08-18.
[32] ESNet, "Energy Sciences Network," https://my.es.net, accessed: 2021-08-18.
[33] Z. Liu, R. Kettimuthu, J. Chung, R. Ananthakrishnan, M. Link, and I. Foster, "Design and evaluation of a simple data interface for efficient data transfer across diverse storage," *ACM Transactions on Modeling and Performance Evaluation of Computing Systems*, vol. 6, no. 1, pp. 1–25, 2021.
[34] Y. Liu, Z. Liu, R. Kettimuthu, N. Rao, Z. Chen, and I. Foster, "Data transfer between scientific facilities–bottleneck analysis, insights and optimizations," in *19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*. IEEE, 2019, pp. 122–131.
[35] Z. Liu, R. Kettimuthu, P. Balaprakash, N. S. Rao, and I. Foster, "Building a wide-area file transfer performance predictor: An empirical study," in *International Conference on Machine Learning for Networking*. Springer, 2018, pp. 56–78.
[36] R. Kettimuthu, Z. Liu, D. Wheeler, I. Foster, K. Heitmann, and F. Cappello, "Transferring a petabyte in a day," *Future Generation Computer Systems*, vol. 88, pp. 191–198, 2018.
[37] A. Sergeev and M. Del Balso, "Horovod: fast and easy distributed deep learning in TensorFlow," *arXiv preprint arXiv:1802.05799*, 2018.
[38] M. Wilamowski, D. A. Sherrell, G. Minasov, Y. Kim, L. Shuvalova, A. Lavens, R. Chard, N. Maltseva, R. Jedrzejczak, M. Rosas-Lemus, N. Saint, I. T. Foster, K. Michalska, K. J. F. Satchell, and A. Joachimiak, "2'-o methylation of RNA cap in SARS-CoV-2 captured by serial crystallography," *Proceedings of the National Academy of Sciences*, vol. 118, no. 21, 2021.
[39] K. Rocki, D. Van Essendelft, I. Sharapov, R. Schreiber, M. Morrison, V. Kibardin, A. Portnoy, J. F. Dietiker, M. Syamlal, and M. James, "Fast stencil-code computation on a wafer-scale processor," in *International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '20. IEEE Press, 2020.
[40] M. Emani, V. Vishwanath, C. Adams, M. E. Papka, R. Stevens, L. Florescu, S. Jairath, W. Liu, T. Nama, and A. Sujeeth, "Accelerating scientific applications with SambaNova reconfigurable dataflow architecture," *Computing in Science & Engineering*, vol. 23, no. 2, pp. 114–119, 2021.
[41] R. Acciarri, C. Adams, C. Backhouse, W. Badgett, L. Bagby, V. Basque, Q. Bazetto, A. Bhanderi, A. Bhat, D. Brailsford *et al.*, "Cosmic background removal with deep neural networks in SBND," *arXiv preprint arXiv:2012.01301*, 2020.
[42] J. Thorpe, Y. Qiao, J. Eyolfson, S. Teng, G. Hu, Z. Jia, J. Wei, K. Vora, R. Netravali, M. Kim *et al.*, "Dorylus: Affordable, scalable, and accurate GNN training with distributed CPU servers and serverless threads," in *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*, 2021, pp. 495–514.
[43] M. Salim, T. Uram, J. T. Childers, V. Vishwanath, and M. E. Papka, "Toward real-time analysis of experimental science workloads on geographically distributed supercomputers," *arXiv preprint arXiv:2105.06571*, 2021.