Performance and Scalability of Distributed Deep Learning

Applicant Institution: University of Texas at El Paso
Street Address: 500 West University Avenue, El Paso, TX 79968
Postal Address: Same as street address

Principal Investigator (PI): Shirley V. Moore
(915) 747- 5054
svmoore@utep.edu

Administrative POC: Danielle Baeza
(915) 747-7940
debaeza@utep.edu

FOA Number: DE-FOA-0002391
DOE/Office of Science Program Office: Advanced Scientific Computing Research
Subprogram: Computer Science
DOE/SC Program Office Technical Contact: Dr. Hal Finkel, Hal.Finkel@science.doe.gov

PAMS Pre-application tracking number: PRE-0000033402

Table of Contents

1 Introduction **1**
- 1.1 Program Relevance 1
- 1.2 Background 1
- 1.3 Related Work 3

2 Project Objectives **3**

3 Proposed Research and Methods **4**
- 3.1 Deep Learning HPC Benchmarks 4
- 3.2 Performance Models of Deep Learning 5
  - 3.2.1 Modeling Memory Requirements 5
  - 3.2.2 Modeling Data Movement 6
  - 3.2.3 Modeling Communication Requirements 6
- 3.3 Communication Analysis 7
- 3.4 Analysis of Coupled Simulation and Deep Learning Workflows 8

4 UTEP/PNNL Collaboration **9**
- 4.1 Research Team 10

5 Timetable of Activities **10**

Appendix 1 – Bibliography and References Cited **12**

Appendix 2 – Facilities and Other Resources **19**

Appendix 3 – Equipment **20**

Appendix 4 – Data Management Plan **22**

Appendix 5 – Promoting Inclusive and Equitable Research (PIER) Plan **23**

Appendix 6 – Other Attachments **24**

1 Pacific Northwest National Laboratory Letter of Support **24**

PROJECT NARRATIVE

1 Introduction

1.1 Program Relevance

The DOE AI for Science report highlights the need to "develop software environments to enable AI capabilities to seamlessly integrate with large-scale HPC models" [70]. The goal is to improve the performance and quality of simulation models using machine learning surrogates. Machine learning is good at integrating multi-modal and multi-fidelity data in order to reveal correlations between features. AI has already been applied successfully in different aspects of fundamental sciences, including information science, mathematics, medical science, materials science, geoscience, life science, physics, and chemistry [83]. However, naïve use of machine learning can result in ill-posed problems and generate non-physical predictions [56]. The AI for Science report further states that incorporating domain knowledge is a distinguishing feature of AI within the DOE mission. According to the report, DOE is unique in the breadth of diverse datasets and representations produced by simulation, experimental, and observational data. However, such data can be sparse, noisy, and unbalanced. Physics-based information in the form of partial differential equations, boundary conditions, and constraints can regularize a machine learning problem so that it can learn from sparse and noisy data [56].

While pre-trained deep learning models are available for applications in image and natural language processing, surrogate models for HPC simulations usually need to be built and trained from scratch. Developing and training these surrogate models can be extremely demanding with respect to computational, memory, and communication resources. The leading machine learning frameworks (e.g., PyTorch, TensorFlow, DeepSpeed) have been developed for video, image, and language processing and not for processing scientific data. There are numerous challenges in coupling these frameworks with scientific software systems. The AI for Science report points out the need for efficient mapping of learning systems onto given hardware. Understanding workload characteristics of scientific deep learning will help systems designers and maintainers with system configuration and resource management and with planning future procurements. Performance modeling and analysis of coupled machine learning and simulation workflows will assist developers in configuring their applications and mapping them to the underlying hardware in an efficient manner. Performance analysis of scientific deep learning will help developers identify and fix bottlenecks that inhibit performance and scaling.

1.2 Background

There are a variety of different types of deep neural networks (DNNs). Convolutional neural networks (CNNs) are used to identify patterns and objects in an input set, usually a set of images. A CNN is constructed from different types of layers, including convolution layers, pooling layers, and fully connected layers. A recurrent neural network (RNN) is used to analyze sequential or time series data. In an RNN, connections between nodes can create a cycle, so that they can use information from prior inputs to influence the current input and output, thus giving the RNN a "memory". A graph neural network (GNN) can be represented as a graph. GNNs use pairwise message passing, meaning that the graph nodes upate their representations by exchanging data with their neighbors. A transformer neural network model uses the concept of self-attentions, which differentially weights the significance of different parts of the input data.

A neural network is typically organized into layers, with weights on the connections between nodes in different layers. An optimizer such as stochastic gradient descent (SGD) is used to train deep learning models. The goal is to find a set of weights that minimizes the loss function that compares the target and predicted output values of the network. First forward propagation is used to propagate an input sample through each layer of the neural network to produce an output. A back propagation algorithm is used to update the weights in which the calculation of the gradient of the error function with respect to the weights proceeds backwards through the network. The batch size is the number of input samples to process

before updating the model weights. The number of epochs is how many times the training algorithm will go through the entire training set.

Several deep learning frameworks have been developed to support training and inference, including TensorFlow [4], PyTorch [16], DeepSpeed [7], and MXNet [1]. The Cray Programming Environment (PE) Deep Learning (DL) plugin [3] will be available on the Frontier exascale computer at OLCF. Both the TensorFlow and PyTorch APIs can be used with the plugin. C/C++ and Python3 interfaces are also available. The plugin supports synchronous and asynchronous data parallel training. The plugin has a custom all_reduce for gradient aggregation that is optimized for deep learning workloads and overlaps computation with communication.

The state-of-the art approach for training large DNNs on current GPUs is to use mixed precision training. Parameters and activations are stored in 16 bits to enable the use of high throughput tensor cores. FP32 is the default data type for deep learning frameworks TensorFlow and PyTorch. However, TensorFlow users can make use of the Keras mixed-precision API, and PyTorch users can use PyTorch automatic mixed precision (AMP) to reduce runtime and memory footprint on modern GPUs such as NVIDIA A100 and AMD Instinct MPI250X. Although the computations are done using 16 bits, parameters are stored in FP32 for numerical stability. A mixed precision optimizer also keeps an FP32 copy of optimizer states.

A number of approaches have been proposed for scaling up deep learning and for overcoming the memory limits of a single GPU. Although the amount of GPU memory has been increasing – for example, the NVIDIA A100 GPU has 80GB of high bandwidth memory – it is still not enough for handling the increasingly large sizes of deep learning models that have grown to have billions of parameters. Model accuracy improves with model size, and it is shown in [41] that larger models can be more resource efficient to train that smaller models for a given accuracy target. The most straightforward approach to parallelism is the form of data parallelism that allows the use of multiple GPUs but requires the entire model to fit on each GPU. This is the type of data parallelism used by TensorFlow [8] and PyTorch [44]. Basic data parallelism runs out of memory for models with more than 1.4 billion parameters. Possible solutions include model parallelism, pipeline parallelism, hybrid parallelism that combines data parallelism with model parallelism, and offloading memory to the CPU [26]. However, these types of parallelism change the communication requirements. Model parallelism splits the model vertically and partitions the computation and parameters in each layer across devices and/or nodes. This creates layer interdependencies that generate communication. For example, fully connected layers incur all-to-all communication as opposed to allreduce in data parallelism, and convolutional layers may also not scale efficiently with model parallelism, depending on the input dimensions [26]. Pipeline parallelism splits a model by layer and runs each layer on a difference device [40]. Micro-batching must be used to hide pipeline bubble and this can make some model functionality difficult to implement and can affect convergence. CPU offloading can result in a significant amount of training time spent on data transfers between CPU and GPU. All of these solutions also require changes to the model code.

Recent approaches such as DeepSpeed ZeRO [60]and FairScale's Fully Sharded Data Parallel (FSDP) [54] shard a model's parameters, gradients and optimizer states across data parallel workers, thus reducing memory consumption, while still maintaining the simplicity of data parallelism. This does however increase the communication overhead. Further developments from DeepSpeed include ZeRO-Offload [64] and ZeRO-Infinity [61]. ZeRO-Offload offloads the optimizer memory and computation from the GPU to the host CPU, and ZeRO-Infinity can exploit all heterogeneous memory of a system, including GPU memory, CPU memory, and non-volatile memory (NVMe). FSDP was made available as a prototype feature in PyTorch 1.11 and remains available as a prototype in Pytorch 2.0 [86], with an automated default sharding policy provided as well as an API for implementing more complex sharding policies. The Colossal-AI [27] deep learning framework uses the PatrickStar chunk-based dynamic memory management described in [31]. PatrickStar organizes the model memory in chunks and dynamically orchestrates the chunks between the GPU and CPU memory.

1.3 Related Work

Previous work on performance modeling of DNNs includes analytical modeling based on computational, memory, and/or communication requirements [40, 59], theoretical derivation of lower bounds for communication in CNNs [29], roofline analysis [79], and application of machine learning to model DNN performance [78]. The modeling in [40] considers various types of deep learning parallelism but models only convolutional neural networks and does not consider runtime issues. An approach to analytical estimation of memory consumption of deep learning models that takes into consideration memory used by both the computation graph and the deep learning framework runtime is proposed in [35].

The PyTorch profiler can be used to measure the time and memory consumption of a deep learning model's operators [46]. The new version of the profiler collects both GPU hardware and PyTorch related information, correlates the two types of information, and tries to detect performance bottlenecks and generate recommendations on how to resolve the bottlenecks. This information from the profiler can be visualized using TensorBoard. The GPU hardware information for NVIDIA GPUs is captured using the NVIDIA CUPTI API [5]. The profiler currently reports timing and flops but not other hardware counter information such as cache and memory statistics. The profiler can show the amount of memory that was allocated and released for the DNN model's operators during execution. The profiler can output a .json trace file and the sequence of profiled operators and CUDA kernels can be visualized using the Chrome trace viewer. TensorBoard also has a trace view that shows the timeline of profiled operators and GPU kernels. The PyTorch profiler with the TensorBoard plugin supports profiling distributed data parallelism with NCCL/GLOO as the backend. The TensorBoard communication view shows computation/communication ratio and overlapping, efficiency of communication in terms of data transfer time and synchronizing time, and detailed communication statistics for each worker. AMD, along with other PyTorch codebase developers in the PyTorch Foundation, has updated the ROCm open software with native support for PyTorch on AMD Instinct GPUs, including support for the new PyTorch profiler, moving to stable support as of PyTorch 1.12 [23].

The TensorFlow profiler includes a suite of tools to help users understand, debug, and optimize TensorFlow programs on CPUs, GPUs, and TPUs [14]. The profiler is accessed from the Profile tab in TensorBoard. After collecting data, the summary view breaks down average step time into categories including time spent reading input data, kernel launch time, device-to-device communication time, on-device compute time, and Python overhead. The profiler reports when a model is input bound and recommends other tools to use to locate and resolve model performance bottlenecks. The trace view displays a timeline that shows the durations of operations that were executed by the TensorFlow model and which part of the system (host or device) executed an operations. The GPU kernel stats display shows performance statistics and the originating operation for every GPU accelerated kernel. The memory profile tool monitors and reports the memory usage of the device and can display a memory timeline graph. The TensorFlow profiler supports multiple GPU profiling for single host systems only.

Deep Learning Profiler (DLProf) from NVIDIA is a wrapper around Nsight Systems that correlates profile timing data and kernel information with a deep learning model [9]. DLProf has multi-GPU support on a single node and supports multiple deep learning frameworks. DLProf reports on tensor core eligibility and usage. It has full support for TensorFlow and PyTorch and provides a simple mode that can be used with any deep learning framework.

2 Project Objectives

The proposed work addresses the objective of intelligent, adaptive resource management and efficient use of heterogeneous computing technologies for parallel and distributed deep learning applications. Deep learning is increasingly being used to augment traditional modeling and simulation in a number of scientific and engineering application areas. While GPUs are currently the main workhorse for parallel deep learning, domain-specific accelerators such as TPUs are claiming some of the market and predicted to grow in

PREVIEW Date: Apr 11, 2023 Workspace ID: WS01056886 Funding Opportunity Number: DE-FOA-0002931

importance. Because of increasingly large and complex datasets for which the computational and memory demands exceed the resources of a single CPU or GPU, various forms of parallelism are used to scale up deep learning. While versions of popular deep learning tools implement parallelism and some performance analysis and optimization tools and guidelines exist, parallelization strategies and performance tuning are based largely on empirical and trial-and-error methods rather than model-based methods. Also, there is no standard portable methodology and tool suite to analyze performance of different deep learning frameworks across multiple processor architectures (e.g., different vendor GPUs, TPUs) and communication networks.

The proposed research aims to provide the users and developers of deep learning frameworks, as well as system administrators, with the tools needed to analyze, optimize, and scale deep learning models on high performance computing platforms. This project does not address strategies for increasing the accuracy of DNN models, other than assessing how such strategies affect performance, but rather focuses on how efficiency and scalability of a given approach can be evaluated and optimized. The technical approach described below encompasses performance modeling, communication analysis, and instrumentation and analysis of coupled machine learning/simulation workflows.

3 Proposed Research and Methods

3.1 Deep Learning HPC Benchmarks

MLPerf HPC suite's Cosmoflow, DeepCAM, and OpenCatalyst (plus other workflows from collaborators) will be used to characterize the performance of enhanced and/or novel machine learning frameworks running on state-of-the-art hardware.

The proposed project will use a set of scientific machine learning benchmarks and applications to motivate, design, and evaluate the methodologies and tools to be developed. The MLPerf HPC benchmark suite includes scientific applications that use ML, especially deep learning, at HPC scale [32]. We intend to use the MLPerf HPC benchmarks to measure and quantify characteristics unique to HPC deep learning applications. We will also use them to evaluate performance and scalability using the deep learning frameworks available on DOE HPC systems, as well as to validate our performance models, which are discussed in section 3.2. We plan to join the MLPerf HPC Working Group and participate in the weekly meetings in order to increase awareness of our research and receive feedback on our results.

The MLPerf HPC suite currently includes Cosmoflow, DeepCAM, and Open Catalyst, with reference implementations available in a Github repository. CosmoFlow is able to process large 3D dark matter distributions and predict cosmological parameters with unprecedented accuracy [50]. CosmoFlow uses a 3D convolutional neural network. The reference implementation is written in TensorFlow with the Keras API and uses Horovod for distributed training [2]. The input dataset used for the benchmark comes from simulations run by the ECP ExaLearn project and is hosted at NERSC.

The goal of DeepCAM is to use deep learning to detect, classify, and localize extreme weather patterns [43]. DeepCam uses deep learning methods to extract high-quality, pixel-level segmentation masks of weather patterns. The DeepCAM developers used and evaluated two different neural network models for doing the segmentation. One model is a modification of the Tiramisu network, which is an extension to the ResNet architecture. Tiramisu introduced skip connections between layers to force the network to learn residual corrections to layer inputs. The second network that was evaluated is based on DeepLabv3+, which is an encoder-decoder architecture, with ResNet-50 used as a core. ResNet-50 is a convolutional neural network that is 50 layers deep. The reference implementation for DeepCAM uses PyTorch [6]. The dataset for the benchmark comes from CAM5 simulations and is hosted at NERSC.

The goal of the Open Catalyst project is to use AI to model and discover new catalysts for use in renewable energy storage [13, 88]. New catalyst structures can be tested and evaluated through the use of quantum chemistry simulations, but these simulations have a very high computational cost. Machine learning can potentially provide a way to efficiently approximate the simulation results. The OC20 and

OC22 datasets for Open Catalyst contain 1.3 million molecular relaxations with results from over 260 million DFT calculations. The types of neural network models used in the Open Catalyst project include some graph neural network (GNN) based models (GemNet [36], SpinConv [69], ForceNet [37], Dimenet [42]), a continuous filter CNN (SchNet [67]), and a crystal graph CNN (CGCNN [82]). These neural networks take chemical structures as inputs and predict energy, forces, and positions.

CosmoFlow, DeepCAM, and ResNet-50 are characterized in [38] with respect to memory requirements, data reuse, performance efficiency across single and multiple GPUs, and data movement across memory and storage hierarchies. The authors show that the three models stress or benefit from architectural features differently, even though they all have an image processing foundation. Their work was done before the advent of DeepSpeed. We plan to carry out performance modeling and analysis of all three MLPerf HPC benchmarks with recent deep learning frameworks on HPC systems with newer GPUs. We will also evaluate the usefulness of on-node NVMe burst buffers in light of technologies such as ZeRO-Infinity [61]. In addition to the MLPerf HPC benchmarks, we plan to obtain additional benchmark and application codes through PNNL contacts.

3.2 Performance Models of Deep Learning

Performance modeling will assist deep learning users in selecting parallelization strategies and configurations and in determining opportunities for performance optimization.

Here performance model means a model of the training efficiency on the underlying hardware, not of the accuracy of the DNN. Performance tuning guides generally suggest that developers try different batch sizes, learning rates, parallelization strategies, etc., to find the most efficient configurations. Additional runs for these performance experiments add to the overall costs of deep learning. A performance model that illustrates how the DNN is mapped onto the underlying hardware resources can reduce the search space for efficient solutions and thus reduce the number of experiments that need to be run. The proposed approach for this project will build on previous work on analytical modeling and make use of hardware counter based measurements to parameterize the analytical models and validate their predictions for different deep learning configurations.

3.2.1 Modeling Memory Requirements

The capability of modeling and predicting memory requirements for DNNs is important for enabling users of deep learning to configure their applications and know whether their problem can fit within the constraints of their platform or allocation. If adequate memory resources are not provisioned, the deep learning job can fail from out-of-memory (OOM) errors, and this is indeed the most common reason for deep learning jobs to fail [35]. Predicting memory consumption of deep learning models has become increasingly difficult because of the growing variety and complexity of how parallelism and memory management are handled by different deep learning frameworks.

Memory consumption of DNNs depends on many factors, including the neural network architecture, the optimizer being used, input size, and batch size. Large batch sizes can improve learning performance but can also significantly increase memory consumption.

The memory consumption for forward propagation consists of memory used to store the parameters of the network (the weights and biases) and memory used for the activations. Memory consumption for the parameters remains constant, but memory consumption for the activations depends on the input item size (e.g., image size) and the batch size. Given the architecture of a neural network, it is straightforward to compute the number of parameters. Given an assumption about the numerical precision used to store the parameters, one can calculate the memory consumption for the parameters. For a convolutional neural network (CNN), given the size of an input array (e.g., a 224 x 224 image) and the number of filters for each layer, the memory consumption for the activations can be calculated [38]. The activation memory of a transformer-based model is proportional to $number of transformer layers \times hidden dimensions \times$

$sequencelength \times batchsize$ [64]. Activation memory can grow quite large for large models, even with activation checkpointing and recomputation. For example, a GPT-like model with 100 billion parameters requires around 60 GB of memory for a batch size of 32, even with activation checkpointing.

For training, the back propagation phase requires storing the gradients for the weights. This doubles the memory consumption for the parameters. If an optimizer is used, then the memory consumption is increased further. If a trained DNN is being used only for inference, then only the forward propagation step is needed, and only parameters and activations need to be stored in memory, and since activations can be discarded when the forward propagation is done with the layer, only the activations for two consecutive layers need to be held in memory temporarily. Thus, memory needed for inference is much less than for training.

Additional memory consumption is due to memory use by the framework runtime, including allocation policies, memory fragmentation, temporary buffers, internal usage (e.g., CUDA context), implementation choices, and operator scheduling [35]. This type of memory consumption is the most difficult to model and predict, since it is hidden inside the runtime. PatrickStar collects memory statistics for this type of memory consumption during a warm-up iteration. DNNMem estimates the account of memory required for CuDNN workspace, CUDA context, temporary tensors, internal tensor fragmentation, and allocation reservations, depending on the policies used by the framework runtime, but these estimates can have errors on the order of 40 to 80%.

We plan to develop a basic memory consumption model for each of the different types of neural networks. This model will be able to be parameterized by the specific DNN model characteristics and the optimizer being used, as well as by input and batch sizes. The memory consumption model will then be combined with implementation characteristics including numerical precisions and the specific deep learning framework being used. We will conduct memory profiling for the different deep learning frameworks to determine the memory consumption for the underlying runtime as accurately as possible. We will validate our memory consumption model by using the available memory profiling tools with our set of deep learning benchmarks.

3.2.2 Modeling Data Movement

An issue related to memory consumption is that of data movement. The research in [38] showed that CosmoFlow moved an order of magnitude more data from high bandwidth memory (HBM) than was necessary. The research in [39] uses custom encoding/decoding strategies with compression/decompression fused with computational operators to optimize preprocessing of scientific images and results in reducing data movement significantly and improving performance by up to ten times. Roofline modeling can be used to determine whether deep learning applications are bound by computation or data movement [84]. Although deep learning models are generally computation intensive, they can be bound by data movement, especially if the training set does not fit into memory and has to be repeatedly accessed from local or even network storage. We will use roofline models of the GPU kernels of deep learning benchmarks to determine whether they are bound by data movement. In the case of memory bound kernels, we will derive lower bounds for data movement for our benchmark applications and compare with measurements of actual data movement. This analysis will help provide insights into how data movement can be reduced across the memory and storage hierarchy.

3.2.3 Modeling Communication Requirements

The communication operations required for deep learning depend on the type of neural network and the parallelization strategy. As an example with respect to the type of neural network, specialized communication algorithms can be used to reduce the communication overhead for graph neural networks [76]. With data parallel training, the main communication operation is an all-reduce that averages gradients across GPUs. Each GPU generates a set of gradients during back propagation that must be globally reduced before updating the model parameters. Asynchronous updating can be used to reduce the communication

PREVIEW Date: Apr 11, 2023 Workspace ID: WS01056886 Funding Opportunity Number: DE-FOA-0002931

overheads but this can affect convergence. Communication operations for model parallelism depend on the type of layer and the way the model is decomposed and can include all-to-all, all-reduce, all-gather, and halo exchange [26]. Implementations of the collective communication operations typically use ring-based or tree-based algorithms. The algorithm can be modeled analytically, resulting in an expression containing a latency term and a bandwidth term. Several algorithms have been developed for the all-reduce operation, and the most efficient algorithm depends on the network topology, the number of processes, and the message size. Rabenseifner's algorithm for all-reduce combines reduce-scatter with all-gather and is bandwidth optimal. The low-level communication substrate, message scheduling, and topology mapping strategies can also affect communication performance.

We will develop analytical communication models for the communication operations used by the various deep learning parallelization strategies and frameworks and parameterize these models with communication measurements. The resulting models will assist users in selecting the best parallelization strategy and configuration and in assessing whether the communication performance they are seeing is within the bounds predicted by the model. The models will also enable exploration of alternative communication strategies for developers of machine learning frameworks.

3.3 Communication Analysis

Developing a unified approach to collecting communication traces will enable portable cross-platform analysis of communication performance, including communication analysis for coupled simulation and deep learning workflows.

Communication profiling can show the amount of time spent performing communication, but when communication overhead grows, such profiling does not give much insight into the cause of increased communication overhead. Users of distributed deep learning frameworks sometimes observe that the proportion of time spent in communication increases as they scale up their model. The increase may be due to unnecessary communication that is an artifact of the deep learning framework, use of inefficient collective communication algorithms, or load imbalance that manifests as increased time spent in communication. Without tools to determine the types and amount of communication and to break down communication into lower-level events, users are left guessing at the cause of the increased communication overhead.

Communication backends used for distributed deep learning include NVIDIA Collective Communication Library (NCCL) [12], ROCm Communication Collectives Library (RCCL) [17], Gloo [10], and MPI. NCCL implements multi-GPU and multi-node communication primitives optimized for NVIDIA GPUs and networking. Collective operations provided include all-gather, all-reduce, broadcast, reduce, and reduce-scatter optimized for PCIe and NVLINK interconnects within a node and NVIDIA Mellanox networks across nodes. RCCL is a library of collective communication routines for ROCm supported GPUs that implements the collective operations all-reduce, all-gather, reduce, broadcast, reduce-scatter, gather, scatter, and all-to-all. RCCL supports multiple GPUs on either a single node or across multiple nodes. RCCL has been optimized for througput and latency on platforms using PCIe and xGMI, as well as networking using InfiniBand Verbs or TCP/IP sockets. It can be used with either single-process or multi-process (e.g., MPI) applications. Gloo is a collective communications library for machine learning applications that includes barrier, broadcasts, and all-reduce. Gloo can be used together with NCCL and/or MPI. Transport between nodes can use TCP/IP or InfiniBand. If InfiniBand transport is used, GPUDirect can be used to accelerate cross-node GPU-to-GPU memory transfers.

Horovod is a generic communication library that supports data parallel training across deep learning frameworks including TensorFlow, PyTorch, and MXNet [11]. Horovod is based on MPI concepts such as size, rank, local rank, allreduce, allgather, broadcast, and alltoall. Horovod uses MPI underneath by default but can be configured to use Gloo instead. Horovod also supports NCCL and Intel oneCCL.

Our approach to communication analysis will be to provide tools for collecting and examining communication traces for different deep learning frameworks on a variety of DNN models. The focus initially will be on the use of Horovod with MPI, since Horovod is widely used with DNN frameworks such as Tensorflow and Pytorch on HPC systems. Score-P is a performance measurement infrastructure for profiling and event tracing of parallel applications [18]. It uses the Open Trace Format Version 2 (OTF2) for event traces. Score-P traces can be collected for Horovod by configuring the Score-P binding for Python, and trace file size can be limited by tracing a few epochs. The Score-P Python binding currently only supports MPI and SHMEM multiprocessing. We will work with the Score-P developers, with whom we have collaborated in the past as part of the Virtual Institute - High Productivity Supercomputing (VI-HPS), to extend Score-P communication profiling to other forms of multiprocessing used by deep learning frameworks. Being able to collect profile and hardware counter information and trace communication for coupled simulation and deep learning workflows (discussed in section 3.4) with the HPC community accepted Score-P standard will enable integrated performance analysis of these workflows.

Previous work has shown that contention for hardware resources such as memory, network, and I/O bandwidth and the direct memory access (DMA) engine can lead to load imbalance that can propagate through the computational DAG [45,58]. Fixed resource allocations can lead to idle resources and bottlenecks. Developing a lightweight monitoring infrastructure based on Score-P will enable deep learning users and system administrators to detect and perform root cause analysis of load imbalance and scaling bottlenecks due to resource contention issues.

Trace files can be analyzed in a scalable manner using tools such as Vampir and Scalasca. The communication traces can also be input to a scalable network simulator to evaluate how use of different topology mappings and collective communication algorithms affect performance. This project will develop guidelines for users, including scientists and HPC operators, on how to use these tools to analyze and optimize communication performance associated with large scale deep learning frameworks.

3.4 Analysis of Coupled Simulation and Deep Learning Workflows

Coupling of simulation and deep learning will bring new challenges in resource management, performance optimization, and scalability. A unified instrumentation and analysis framework for the coupled workflows will help address these challenges.

As discussed in the DOE AI for Science report [70], machine learning approaches can help scientists explore data to gain new insights leading to scientific discovers. Machine learning is especially promising for accelerating quantum chemistry calculations and discovery of new materials, with successes already reported [80–82, 87]. Surrogate modeling has been successfully used to accelerate demanding applications in fusion science. As reported in [65], MIT scientists started with a set of computationally intensive local calculations run with the full-physics, first-principles CGYRO code and then fit a surrogate model which was used to explore and optimize a search within the parameter space. The coupling led to a converged solution with a minimal number of expensive gyrokinetic simulations without compromising accuracy. Physics-informed machine learning in biomedicine has been successfully used to model cardiovascular flows and cardiac activation mapping [56].

PI Moore has previously worked with the ECP WDMApp team on developing the EFFIS workflow and code coupling framework for the Whole Device Modeling Application (WDMApp) [72]. EFFIS uses TAU [68] and PerfStubs [28] for performance instrumentation and analysis. In the case of coupled execution, the goal is to understand the performance of not only individual components but also the interactions between components. It is also necessary to keep overhead of performance measurement low so as to not distort application performance. PerfStubs is a thin stub interface for instrumenting library and application code. Instrumentation calls are stubs in the form of function pointers that are optionally assigned at runtime. TAU implements a PerfStubs plugin that can collect profile and trace data using Score-P. The performance

data for coupled codes is made available in a unified way through use of a consistent schema. The combined performance data enables diagnosis of performance problems such as load imbalance, under-utilization of resources, excessive wait times, and expensive data movement.

We will engage with our colleagues in computational chemistry, specifically the GAMESS ECP team, plasma physics simulation, and biomedical simulation [57] who are interested in combining simulation with deep learning, as well as connections made through PNNL, to solicit requirements for performance analysis of the coupled workflows. We will then design a framework for unified performance monitoring with the goal of collecting data about resource utilization, data movement, and excessive wait times so as to be able to identify and fix bottlenecks in a coupled workflow.

4 UTEP/PNNL Collaboration

PNNL will facilitate a tighter connection between UTEP and DOE ASCR by helping to build a systems capability research agenda at UTEP, provide access to PNNL staff and compute resources, and build broader strategic partnerships with PNNL and its partners.

As of 2021, the Hispanic population makes up 17% of the working population but only 7% of STEM employment [34]. Enrolling close to 24,000 students, UTEP plays an essential role in promoting this population group in STEM, boasting a student body that is 84% Hispanic [20]. UTEP's computer science department enrolled 1307 students in 2022, 1054 of whom are Hispanic [21]. Of the students enrolled for the 2022-2023 academic year, the faculty will advise 72 masters and 39 Ph.D. students (62.5% and 30.8% Hispanic, respectively). While UTEP's faculty have gained recognition with two prestigious early career awards from NSF [22], faculty has had limited engagement with DOE ASCR. Several opportunities exist to establish a productive relationship between UTEP and DOE ASCR, including building a systems capability research agenda, reducing barriers to access to PNNL staff and compute resources, and building broader strategic partnerships with PNNL and its partners.

While UTEP boasts research across various disciplines within computer science, such as data analytics, cyber security, robust autonomic systems, and ML, a broader research agenda surrounding high performance computing (HPC) has yet to be developed. The proposed research bridging HPC and ML is an essential first step while addressing a priority research direction underscored by the DOE. In its 2019 SciML workshop report [25], ASCR highlighted the need for robust machine learning, identifying the necessity to understand the quality of an ML result in the context of its computational performance and data preparation and management. Further, the proliferation of funding for AI/ML can be seen across the various DOE efforts and announcements [51–53, 71]. As a DOE laboratory, PNNL's successful work in AI/ML can be attributed partly to its clear focus and strategic investment in the field. PNNL touts scalable machine reasoning for scientific discovery to realize machine reasoning capabilities and apply them to accelerate scientific discovery as one of six laboratory objectives [15]. Leveraging this work as a springboard, PNNL intends to work with UTEP on a systems/HPC-based research agenda. Concretely, this will include working with faculty to identify research opportunities in ML/HPC of interest to DOE clients. PNNL will work with UTEP to provide access to internal brown bags and workshops to understand ASCR's needs and funding opportunities better. The goal is to build joint capabilities useful to DOE.

One of the challenges to performing research in HPC is accessing cutting-edge computing platforms. While PNNL has several (Nvidia and AMD) GPU-based clusters appropriate for HPC/ML research, through projects like CENATE [47], PNNL has novel architectures like SambaNova [66] and NextSilicon [63]. These resources will be made available to UTEP to facilitate the project milestones and its larger research agenda. In addition to the computing resources and their support staff, UTEP faculty will have access to AI and HPC experts at PNNL. This broader engagement will include seminars, brown bags, workshops, and on-site visits, which will help build UTEP's research capability and develop collaborations among various experts at the lab.

Further, a noted challenge at UTEP is building sustainable research groups around systems/HPC fields. By focusing on matching researchers at PNNL with UTEP faculty and students, we will increase interest and awareness of the challenges, priorities, and opportunities at DOE ASCR. We will prioritize exposing existing ASCR research projects to UTEP students to promote both internships at PNNL and research assistance at UTEP through seminars and guest lectures. This effort will span all collegiate levels to build sustainable research groups at UTEP.

In addition to the various research projects around HPC and ML, our effort will leverage a recent institutional partnership between PNNL and UTEP [19]. This partnership aims to increase internships and employment, facilitate joint appointments, promote board/committee memberships, explore research investments, and submit joint proposals. Having only started at the beginning of this fiscal year, the effort has established five internships, four join appointees, and one board membership. Such progress has been in material science, cyber security, and earth science. Our proposed work will leverage the connections made via this partnership to facilitate an HPC/ML research agenda targeting the specific needs of DOE ASCR. By partnering with existing efforts, we can better steward DOE investments in tackling the DEI challenges in HPC.

Our metrics of success will include joint papers, engagement opportunities, PNNL internships, new UTEP student research assistants, and future proposals. In addition, these metrics will extend beyond this project but remain focused on DOE ASCR to demonstrate clear attribution.

4.1 Research Team

Our research team is composed of experts in the areas of modeling, performance analysis, and machine learning from both UTEP and PNNL. The UTEP members of the team, Moore and Tosh, are involved with research and teaching in the areas of computer architecture, operating systems, networking, high performance computing, and distributed machine learning. They co-supervise students working on network slicing and orchestration of distributed machine learning applications for anomaly detection in cyber physical systems. Tosh is a recent recipient of the NSF early career award. Together, Tosh and Moore have been working on a collaborative DOE project to design and develop quality-of-service aware distributed edge computing architecture that can accommodate different machine learning variants including federated learning, and distributed machine learning models to detect component health related abnormalities in power generation cyber-physical environments. This has resulted multiple peer-reviewed publications [48, 49].

The PNNL members of the team, Suetterlein and Manzano, have previously collaborated with Moore on performance modeling for ECP [24]. Suetterlein has been part of the Exalearn ECP project and has expertise in HPC modeling, distributed memory models, asynchronous runtime systems, and grid computing [33, 62, 73–75]. Manzano's research areas includes domain specific accelerators for deep learning, power analysis, modeling, application-based resilience, and distributed simulators [30, 55, 77, 85]. Our team brings together the requisite expertise in high performance computing, networking, performance analysis, and machine learning.

5 Timetable of Activities

Figure 5.1 shows the sub-tasks that each of the thrusts will take to accomplish the proposal objectives. Figure 5.2 is a graphical timeline representing each thrust's sub-tasks and their duration across the three years performance period.

3.1 Deep Learning HPC Benchmarks
- (a) Develop performance models for MLPerf HPC benchmarks
- (b) Collect and analyze performance data across machine learning frameworks and GPU architectures
- (c) Evaluate use of NVMe burst buffers
- (d) Model and analyze selected PNNL DL applications

3.2 Performance Models of Deep Learning
- (a) Develop basic analytical models
- (b) Add parameterization to capture implementation details
- (c) Test models on benchmarks
- (d) Refine models as needed

3.3 Communication Analysis
- (a) Implement Score-P tracing for Horovod+MPI
- (b) Extend Score-P to additional multiprocessing libraries
- (c) Trace and analyze communication for MLPerf HPC benchmarks
- (d) Trace and analyze communication for coupled ML/simulation workflows

3.4 Analysis of Coupled Simulation and Deep Learning Workflows
- (a) Solicit requirements for performance analysis of coupled ML/simulation workflows
- (b) Develop workflow analysis framework using PerfStubs and TAU
- (c) Instrument coupled computational chemistry application
- (d) Instrument coupled fusion application
- (e) Instrument coupled biomedical application

4 UTEP/PNNL Collaboration
- (a) Define broader collaborative research agenda
- (b) Identify strategic partnerships for internships and collaborations
- (c) PNNL/UTEP Seminar Series
- (d) Guest Lecture

Figure 5.1: Milestones and Tasks.

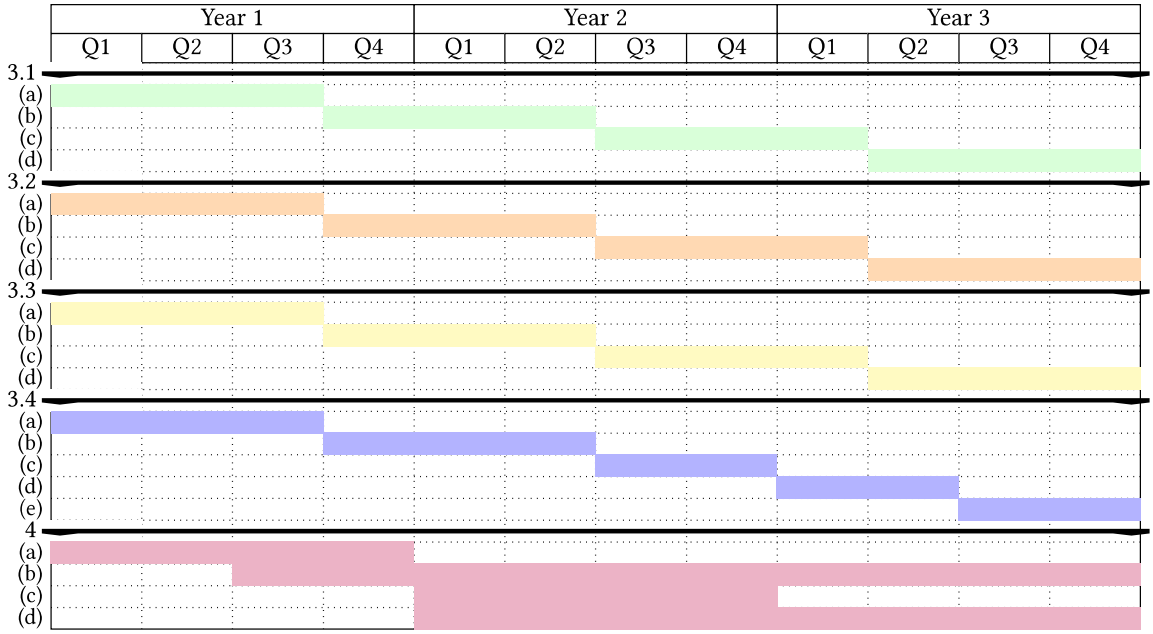

Figure 5.2: Timeline of activities broken down by quarter.

APPENDIX 1: BIBLIOGRAPHY AND REFERENCES CITED

[1] Apache MXNet: A flexible and efficient library for deep learning. https://mxnet.apache.org. Online; accessed April 10, 2023.

[2] CosmoFlow TensorFlow Keras benchmark. https://github.com/mlcommons/hpc/tree/main/cosmoflow. Online: accessed April 9, 2023.

[3] Cray Programming Environment (PE) Deep Learning (DL) Plugin - Cray Distributed Training Framework. https://support.hpe.com/hpesc/public/docDisplay?docId=a00114874en_us&docLocale=en_US&page=Deep_Learning_with_the_Cray_Programming_Environment_DL_Plugin.html#distributed-deep-learning. Online; accessed April 10, 2023.

[4] Create production-grade machine learning models with TensorFlow. https://tensorflow.org. Online; accessed April 10, 2023.

[5] CUPTI: The API reference guide for CUPTI, the CUDA profiling tools interface. https://docs.nvidia.com/cupti/index.html. Online; accessed April 10, 2023.

[6] Deep learning climate segmentation benchmark. https://github.com/mlcommons/hpc/tree/main/deepcam. Online, accessed April 9, 2023.

[7] DeepSpeed. https://deepspeed.ai. Online; accessed April 10, 2023.

[8] Distributed training with TensorFlow. https://www.tensorflow.org/guide/distributed_training. Online: Accessed April 8, 2023.

[9] DLProf user guide. https://docs.nvidia.com/deeplearning/frameworks/dlprof-user-guide/index.html. Online; accessed April 10, 2023.

[10] Gloo. https://github.com/facebookincubator/gloo. Online; accessed April 10, 2023.

[11] Horovod documentation. https://horovod.readthedocs.io/en/stable/. Online; accessed April 10, 2023.

[12] NVIDIA NCCL. https://developer.nvidia.com/nccl. Online; accessed April 10, 2023.

[13] Open Catalyst project. https://opencatalystproject.org/. Online; accessed April 9, 2023.

[14] Optimize TensorFlow performance using the profiler. https://www.tensorflow.org/guide/profiler. Online; accessed April 10, 2023.

[15] Pacific Northwest National Laboratory: Lab Objectives: Scalable Machine Reasoning for Scientific Discovery. https://www.pnnl.gov/lab-objectives.

[16] PyTorch 2.0 now available. pytorch.org. Online; accessed April 10, 2023.

[17] RCCL - ROCm communication collectives library. https://github.com/ROCmSoftwarePlatform/rccl. Online; accessed April 10, 2023.

[18] Score-P: Scalable performance measurement infrastructure for parallel codes. https://www.vi-hps.org/projects/score-p/. Online; accessed April 10, 2023.

[19] UTEP / PNNL parternship. https://www.pnnl.gov/projects/utep/partnership. Online: October 2022.

[20] UTEP at a Glance. https://www.utep.edu/initiatives/at-a-glance/.

[21] UTEP college of engineering: Computer science data fact sheet. https://www.utep.edu/cs/about/Enrollment-and-Graduation-Data.pdf. Online: data created for Fall 2022.

[22] UTEP computer science: About us. https://www.utep.edu/cs/about/index.html.

[23] AMD. Democratizing AI with PyTorch Foundation and ROCm support for PyTorch. https://pytorch.org/blog/democratizing-ai-with-pytorch/, 2023. Online; accessed April 10, 2023.

[24] B. Austin, R. Bair, K. Barker, A. Cabrera, A. Chien, N. Ding, J. Firoz, K. Ibrahim, J. Manzano, V. Morozov, T. Nguyen, L. Oliker, J. Suetterlein, L. Tang, J. Vetter, S. Williams, K. Yoshii, and A. Young. Hardware evaluation analytical modeling and node simulation: Benefits of tighter gpu integration. Technical report, Pacific Northwest National Laboratory, 9 2021.

[25] N. Baker, F. Alexander, T. Bremer, A. Hagberg, Y. Kevrekidis, H. Najm, M. Parashar, A. Patra, J. Sethian, S. Wild, K. Willcox, and S. Lee. Workshop report on basic research needs for scientific machine learning: Core technologies for artificial intelligence. February 2019.

[26] T. Ben-Nun and T. Hoefler. Demystifying parallel and distributed deep learning: An in-depth concurrency analysis. *ACM Comput. Surv.*, 52(4), aug 2019.

[27] Z. Bian, H. Liu, B. Wang, H. Huang, Y. Li, C. Wang, F. Cui, and Y. You. Colossal-AI: A unified deep learning system for large-scale parallel training. *arXiv preprint arXiv:2110.14883*, 2021.

[28] D. Böhme, K. A. Huck, J. Madsen, and J. Weidendorfer. The case for a common instrumentation interface for HPC codes. In *IEEE/ACM International Workshop on Programming and Performance Visualization Tools, ProTools@SC 2019, Denver, CO, USA, November 17, 2019*, pages 33–39. IEEE, 2019.

[29] A. Chen, J. Demmel, G. Dinh, M. Haberle, and O. Holtz. Communication bounds for convolutional neural networks. In *Proceedings of the Platform for Advanced Scientific Computing Conference*, PASC '22, New York, NY, USA, 2022. Association for Computing Machinery.

[30] M. Erez, S. Krishnamoorthy, G. Kestor, J. Manzano, L. Song, O. Subasi, B. Mutlu, S. Amarasinghe, M. Carbin, M. Rinard, P. Sadayappan, G. Agrawal, S. Mahlke, and G. Gopalakrishnan. Aedam: Whole program adaptive error detection and mitigation (final report). Technical report, Pacific Northwest National Laboratory, 5 2021.

[31] J. Fang, Z. Zhu, S. Li, H. Su, Y. Yu, J. Zhou, and Y. You. Parallel training of pre-trained models via chunk-based dynamic memory management. *IEEE Transactions on Parallel & Distributed Systems*, 34(01):304–315, jan 2023.

[32] S. Farrell, M. Emani, J. Balma, L. Drescher, A. Drozd, A. Fink, G. C. Fox, D. Kanter, T. Kurth, P. Mattson, D. Mu, A. Ruhela, K. Sato, K. Shirahata, T. Tabaru, A. Tsaris, J. Balewski, B. Cumming, T. Danjo, J. Domke, T. Fukai, N. Fukumoto, T. Fukushi, B. Gerofi, T. Honda, T. Imamura, A. Kasagi, K. Kawakami, S. Kudo, A. Kuroda, M. Martinasso, S. Matsuoka, H. Mendonça, K. Minami, P. Ram, T. Sawada, M. Shankar, T. S. John, A. Tabuchi, V. Vishwanath, M. Wahib, M. Yamazaki, and J. Yin. Mlperf™ HPC: A holistic benchmark suite for scientific machine learning on HPC systems. In *IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments, MLHPC@SC 2021, St. Louis, MO, USA, November 15, 2021*, pages 33–45. IEEE, 2021.

[33] R. D. Friese, B. O. Mutlu, N. R. Tallent, J. Suetterlein, and J. Strube. Effectively using remote I/O for work composition in distributed workflows. In *Proc. of the 2020 IEEE Intl. Conf. on Big Data*. IEEE Computer Society, December 2020.

[34] R. Fry, B. Kennedy, and C. Funk. STEM jobs see uneven progress in increasing gender, racial and ethnic diversity. `https://www.pewresearch.org/science/2021/04/01/stem-jobs-see-uneven-progress-in-increasing-gender-racial-and-ethnic-diversity/`. Online: accessed April 1, 2021.

[35] Y. Gao, Y. Liu, H. Zhang, Z. Li, Y. Zhu, H. Lin, and M. Yang. Estimating GPU memory consumption of deep learning models. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2020, page 1342–1352, New York, NY, USA, 2020. Association for Computing Machinery.

[36] J. Gasteiger, F. Becker, and S. Günnemann. GemNet: Universal directional graph neural networks for molecules. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 6790–6802. Curran Associates, Inc., 2021.

[37] W. Hu, M. Shuaibi, A. Das, S. Goyal, A. Sriram, J. Leskovec, D. Parikh, and C. L. Zitnick. ForceNet: A graph neural network for large-scale quantum calculations. *CoRR*, abs/2103.01436, 2021.

[38] K. Z. Ibrahim, T. Nguyen, H. A. Nam, W. Bhimji, S. Farrell, L. Oliker, M. Rowan, N. J. Wright, and S. Williams. Architectural requirements for deep learning workloads in HPC environments. In *2021 International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS 2021), St. Louis, MO, USA, November 15, 2021*, pages 7–17. IEEE, 2021.

[39] K. Z. Ibrahim and L. Oliker. Preprocessing pipeline optimization for scientific deep learning workloads. In *2022 IEEE International Parallel and Distributed Processing Symposium, IPDPS 2022, Lyon, France, May 30 - June 3, 2022*, pages 1118–1128. IEEE, 2022.

[40] A. N. Kahira, T. T. Nguyen, L. B. Gomez, R. Takano, R. M. Badia, and M. Wahib. An oracle for guiding large-scale model/hybrid parallel training of convolutional neural networks. In *Proceedings of the 30th International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '21, page 161–173, New York, NY, USA, 2021. Association for Computing Machinery.

[41] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei. Scaling laws for neural language models. *CoRR*, abs/2001.08361, 2020.

[42] J. Klicpera, J. Groß, and S. Günnemann. Directional message passing for molecular graphs. *CoRR*, abs/2003.03123, 2020.

[43] T. Kurth, S. Treichler, J. Romero, M. Mudigonda, N. Luehr, E. Phillips, A. Mahesh, M. Matheson, J. Deslippe, M. Fatica, Prabhat, and M. Houston. Exascale deep learning for climate analytics. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, SC '18. IEEE Press, 2018.

[44] S. Li, Y. Zhao, R. Varma, O. Salpekar, P. Noordhuis, T. Li, A. Paszke, J. Smith, B. Vaughan, P. Damania, and S. Chintala. Pytorch distributed: Experiences on accelerating data parallel training. *Proc. VLDB Endow.*, 13(12):3005–3018, aug 2020.

[45] J. Liu, B. Nicolae, and D. Li. Lobster: Load Balance-Aware I/O for Distributed DNN Training. In *ICPP '22: The 51st International Conference on Parallel Processing*, Bordeaux, France, Aug. 2022.

[46] M. Lukiyanov, G. Hua, G. Chauhan, and G. Dankel. Introducing pytorch profiler - the new and improved performance tool. https://pytorch.org/blog/introducing-pytorch-profiler-the-new-and-improved-performance-tool/, 2021. Online; accessed April 10, 2023.

[47] D. Manz and K. Barker. Center for advanced technology evaluation. https://www.pnnl.gov/projects/cenate.

[48] W. Marfo, D. K. Tosh, and S. V. Moore. Network anomaly detection using federated learning. In *MILCOM 2022-2022 IEEE Military Communications Conference (MILCOM)*, pages 484–489. IEEE, 2022.

[49] W. Marfo, D. K. Tosh, and S. V. Moore. Network anomaly detection using federated learning. In *MILCOM 2022-2022 IEEE Military Communications Conference (MILCOM)*, pages 484–489. IEEE, 2022.

[50] A. Mathuriya, D. Bard, P. Mendygral, L. Meadows, J. Arnemann, L. Shao, S. He, T. Kärnä, D. Moise, S. J. Pennycook, K. Maschhoff, J. Sewall, N. Kumar, S. Ho, M. F. Ringenburg, Prabhat, and V. Lee. Cosmoflow: Using deep learning to learn the universe at scale. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage, and Analysis*, SC '18. IEEE Press, 2019.

[51] O. of Science. Department of energy invests $16 million in data intensive scientific machine learning research and analysis. https://www.energy.gov/science/articles/department-energy-invests-16-million-data-intensive-scientific-machine-learning. Published in September 9, 2021.

[52] D. A. Office. Scientific machine learning for modeling and simulations. https://science.osti.gov/-/media/grants/pdf/foas/2020/SC_FOA_0002319.pdf. Published in April 8, 2020.

[53] D. S. Office. Scientific machine learning for complex systems. https://science.osti.gov/ascr/-/media/grants/pdf/foas/2023/DE-FOA-0002958-000001.pdf. Published in January 24, 2023.

[54] M. Ott, S. Shleifer, M. Xu, P. Goyal, Q. Duval, and V. Caggiano. Fully sharded data parallel: faster AI training with fewer GPUs. https://engineering.fb.com/2021/07/15/open-source/fsdp/, 2021. Online: accessed April 8, 2023.

[55] A. Panyala, D. Chavarría-Miranda, J. B. Manzano, A. Tumeo, and M. Halappanavar. Exploring performance and energy tradeoffs for irregular applications: A case study on the tilera many-core architecture. *Journal of Parallel and Distributed Computing*, 104:234–251, 2017.

[56] G. C. Peng, M. Alber, A. Buganza Tepole, W. R. Cannon, S. De, S. Dura-Bernal, K. Garikipati, G. E. Karniadakis, W. W. Lytton, P. Perdikaris, L. Petzold, and E. Kuhl. Multiscale modeling meets machine learning: What can we learn? *Archives of Computatiional Methods of Engineering*, 28(3), 5 2021.

[57] D. F. Puleri, S. Roychowdhury, P. Balogh, J. Gounley, E. W. Draeger, J. Ames, A. Adebiyi, S. Chidyagwai, B. Hernández, S. Lee, S. V. Moore, J. S. Vetter, and A. Randles. High performance adaptive physics refinement to enable large-scale tracking of cancer cell trajectory. In *2022 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 230–242, 2022.

[58] S. Pumma, D. Buono, F. Checconi, X. Que, and W.-c. Feng. Alleviating load imbalance in data processing for large-scale deep learning. In *2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID)*, pages 262–271, 2020.

[59] Qi, E. R. Sparks, and A. Talwalkar. Paleo: A performance model for deep neural networks. In *International Conference on Learning Representations*, 2016.

[60] S. Rajbhandari, J. Rasley, O. Ruwase, and Y. He. Zero: Memory optimization towards training A trillion parameter models. *CoRR*, abs/1910.02054, 2019.

[61] S. Rajbhandari, O. Ruwase, J. Rasley, S. Smith, and Y. He. Zero-infinity: breaking the GPU memory wall for extreme scale deep learning. In B. R. de Supinski, M. W. Hall, and T. Gamblin, editors, *International Conference for High Performance Computing, Networking, Storage and Analysis, SC 2021, St. Louis, Missouri, USA, November 14-19, 2021*, page 59. ACM, 2021.

[62] K. Ranganath, J. Firoz, J. Suetterlein, J. Manzano, A. Marquez, M. Raugas, and D. Wong. Lc-memento: A memory model for accelerated architectures. In X. Li and S. Chandrasekaran, editors, *Languages and Compilers for Parallel Computing*, pages 67–82, Cham, 2022. Springer International Publishing.

[63] E. Raz et al. Nextsilicon. `https://www.nextsilicon.com/`. Published in 2021.

[64] J. Ren, S. Rajbhandari, R. Y. Aminabadi, O. Ruwase, S. Yang, M. Zhang, D. Li, and Y. He. Zero-offload: Democratizing billion-scale model training. In I. Calciu and G. Kuenning, editors, *2021 USENIX Annual Technical Conference, USENIX ATC 2021, July 14-16, 2021*, pages 551–564. USENIX Association, 2021.

[65] P. Rodriguez-Fernandez, N. Howard, and J. Candy. Nonlinear gyrokinetic predictions of sparc burning plasma profiles enabled by surrogate modeling. *Nuclear Fusion*, 62(7):076036, may 2022.

[66] SambaNova Systems. Whitepaper: Accelerated computing with a reconfigurable dataflow architecture. `https://sambanova.ai/wp-content/uploads/2021/04/SambaNova_Accelerated-Computing-with-a-Reconfigurable-Dataflow-Architecture_Whitepaper_English.pdf`. Published in 2021.

[67] K. T. Schütt, P.-J. Kindermans, H. E. Sauceda, S. Chmiela, A. Tkatchenko, and K.-R. Müller. SchNet: A continuous-filter convolutional neural network for modeling quantum interactions, 2017.

[68] S. Shende and A. D. Malony. The TAU parallel performance system. *Int. J. High Perform. Comput. Appl.*, 20(2):287–311, 2006.

[69] M. Shuaibi, A. Kolluru, A. Das, A. Grover, A. Sriram, Z. W. Ulissi, and C. L. Zitnick. Rotation invariant graph neural networks using spin convolutions. *CoRR*, abs/2106.09575, 2021.

[70] R. Stevens, V. Taylor, J. Nichols, A. B. Maccabe, K. Yelick, and D. Brown. AI for Science: Report on the Department of Energy (DOE) town halls on artificial intelligence (AI) for science. 2 2020.

[71] L. Stone. Us department of energy to spend $16m on researching ai tools for hard science, complex systems. `https://aibusiness.com/verticals/us-department-of-energy-to-spend-16m-on-researching-ai-tools-for-hard-science-complex-systems`. Published in September 10, 2020.

[72] E. Suchyta, S. Klasky, N. Podhorszki, M. Wolf, A. D. Adesoji, C. Chang, J. Choi, P. E. Davis, J. Dominski, S. Ethier, I. T. Foster, K. Germaschewski, B. Geveci, C. Harris, K. A. Huck, Q. Liu, J. Logan, K. Mehta, G. Merlo, S. V. Moore, T. S. Munson, M. Parashar, D. Pugmire, M. S. Shephard, C. W. Smith, P. Subedi, L. Wan, R. Wang, and S. Zhang. The exascale framework for high fidelity coupled simulations (EFFIS): enabling whole device modeling in fusion science. *Int. J. High Perform. Comput. Appl.*, 36(1):106–128, 2022.

[73] J. Suetterlein, R. D. Friese, N. R. Tallent, and M. Schram. TAZeR: Hiding the cost of remote I/O in distributed scientific workflows. In *Proc. of the 2019 IEEE Intl. Conf. on Big Data*, pages 383–394. IEEE Computer Society, December 2019.

[74] J. Suetterlein, J. Landwehr, A. Marquez, J. Manzano, K. J. Barker, and G. R. Gao. Verification of the extended roofline model for asynchronous many task runtimes. In *Proceedings of the Third International Workshop on Extreme Scale Programming Models and Middleware*, ESPM2'17, New York, NY, USA, 2017. Association for Computing Machinery.

[75] J. D. Suetterlein, J. Landwehr, A. Marquez, J. Manzano, and G. R. Gao. Extending the roofline model for asynchronous many-task runtimes. In *2016 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 493–496, 2016.

[76] A. Tripathy, K. Yelick, and A. Buluc. Reducing communication in graph neural network training. *SC '20: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 11 2020.

[77] O. Villa, A. Tumeo, S. Secchi, and J. B. Manzano. Fast and accurate simulation of the cray xmt multithreaded supercomputer. *IEEE Transactions on Parallel and Distributed Systems*, 23(12):2266–2279, 2012.

[78] C.-C. Wang, Y.-C. Liao, M.-C. Kao, W.-Y. Liang, and S.-H. Hung. Perfnet: Platform-aware performance modeling for deep neural networks. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems*, RACS '20, page 90–95, New York, NY, USA, 2020. Association for Computing Machinery.

[79] Y. Wang, C. Yang, S. Farrell, Y. Zhang, T. Kurth, and S. Williams. Time-based roofline for deep learning performance analysis. In *2020 IEEE/ACM Fourth Workshop on Deep Learning on Supercomputers (DLS)*, pages 10–19, 2020.

[80] A. D. White. Deep learning for molecules and materials. *Living Journal of Computational Molecular Science*, 3(1):1499, 2021.

[81] T. Williams, K. McCullough, and J. A. Lauterbach. Enabling catalyst discovery through machine learning and high-throughput experimentation. *Chemistry of Materials*, 32(1):157–165, 2019.

[82] T. Xie and J. C. Grossman. Crystal graph convolutional neural networks for an accurate and interpretable prediction of material properties. *Phys. Rev. Lett.*, 120:145301, Apr 2018.

[83] Y. Xu, X. Liu, X. Cao, C. Huang, E. Liu, S. Qian, X. Liu, Y. Wu, F. Dong, C.-W. Qiu, J. Qiu, K. Hua, W. Su, J. Wu, H. Xu, Y. Han, C. Fu, Z. Yin, M. Liu, R. Roepman, S. Dietmann, M. Virta, F. Kengara, Z. Zhang, L. Zhang, T. Zhao, J. Dai, J. Yang, L. Lan, M. Luo, Z. Liu, T. An, B. Zhang, X. He, S. Cong, X. Liu, W. Zhang, J. P. Lewis, J. M. Tiedje, Q. Wang, Z. An, F. Wang, L. Zhang, T. Huang, C. Lu, Z. Cai, F. Wang, and J. Zhang. Artificial intelligence: A powerful paradigm for scientific research. *The Innovation*, 2(4):100179, 2021.

[84] C. Yang, Y. Wang, T. Kurth, S. Farrell, and S. Williams. Hierarchical roofline performance analysis for deep learning applications. In K. Arai, editor, *Intelligent Computing - Proceedings of the 2021 Computing Conference, Volume 2, SAI 2021, Virtual Event, 15-16 July, 2021*, volume 284 of *Lecture Notes in Networks and Systems*, pages 473–491. Springer, 2021.

[85] J. J. Zhang, N. B. Agostini, S. Song, C. Tan, A. Limaye, V. Amatya, J. B. Manzano, M. Minutoli, V. G. Castellana, A. Tumeo, G. Wei, and D. Brooks. Towards automatic and agile AI/ML accelerator design with end-to-end synthesis. In *32nd IEEE International Conference on Application-specific Systems, Architectures and Processors, ASAP 2021, Virtual Conference, USA, July 7-9, 2021*, pages 218–225. IEEE, 2021.

[86] Y. Zhao, R. Varma, C.-C. Huang, S. Li, M. Xu, and A. Desmaison. Introducing pytorch fully sharded data parallel (FSDP) API, 2022. Online, accessed April 8, 2023.

[87] G. Zhou, N. Lubbers, K. Barros, S. Tretiak, and B. Nebgen. Deep learning of dynamically responsive chemical hamiltonians with semiempirical quantum mechanics. *Proceedings of the National Academy of Sciences*, 119(27):e2120333119, 2022.

[88] C. L. Zitnick, L. Chanussot, A. Das, S. Goyal, J. Heras-Domingo, C. Ho, W. Hu, T. Lavril, A. Palizhati, M. Riviere, M. Shuaibi, A. Sriram, K. Tran, B. M. Wood, J. Yoon, D. Parikh, and Z. W. Ulissi. An introduction to electrocatalyst design using machine learning for renewable energy storage. *CoRR*, abs/2010.09435, 2020.

APPENDIX 2: FACILITIES AND OTHER RESOURCES

The University of Texas at El Paso, where PI Moore and Co-PI Tosh are affiliated, will be the main site at which this project will be executed.

1. The **Edge Computing and High Performance Computing** research group led by PI Moore has laboratory space in Chemistry and Computer Science building at UTEP for six students. Lab equipment includes monitors for the students' laptops, a projector, and various single-board computers and microcontrollers used for edge computing experiments. The lab also has access to the group's 32-core Intel IceLake server running Ubuntu 22.04 that is housed in the UTEP Research and Academic Data Center (RADC).

2. **TRUstworthy CYBER systems (TRUCYBER) Laboratory.**
   TRUCYBER lab is directed by Co-I Tosh which is located in Prospect Hall at UTEP. This lab has space of 650 sq. ft and can accommodate 6 students to work. Currently, there are two PhD students and one master student working in the lab. There is still room for couple more students to accommodate who can work on the proposed project. The lab is equipped with gigabit speed Internet, access to high performance computing servers, re-configurable software-defined routers and switches, firewall, and network-attached storage devices. This lab has a dedicated server room that has 6 Dell PowerEdge 740 Rack Servers and 4 Dell Tower (T440) servers. This cluster has been orchestrated to offer local compute facility for various machine learning needs. In addition, the lab has a 65" TV that is used to organize research presentations by students and faculty mentors.

3. **Facilities and Resources availed through PNNL Collaborator.**
   PNNL provides modern laboratory and office space, including standardized conferencing capabilities for video- and tele-conferencing (using Microsoft Teams). Campus conference spaces feature high-definition web cameras and audio devices, large video displays, and built-in table electrical connections. The PNNL research team will have access to all major computer platforms and operating systems identified in this proposal. In addition to the general-purpose networking and desktop computing infrastructure at PNNL, a range of advanced computer architectures is available. In regards to access to high-bandwidth communication, fully redundant Internet connectivity is provided by PNNL's existing 100 Gb/s connections to the U.S. Department of Energy's (DOE) ESnet in Seattle and 20Gb/s to Boise, Idaho. Data transfer nodes provides high-speed file transfer capability to/from PNNL's 5 PB institutional computing storage capability. PNNL uses the Northwest Open Access Network (NOANET) to connect the Richland campus with PNNL facilities in Sequim, Washington and Portland, Oregon. Also, PNNL has access to the Center for Advanced Technology Evaluation (CENATE) which encompasses instrumentation, testbeds, evaluation, and modeling and simulation research areas that primarily focus on workload applications of interest to DOE.

   Thus, combinedly all the labs have enough working space to accommodate the recruited students to conduct collaborative cutting-edge research, and arrange virtual events to work in tandem with PNNL researchers.

Other Supports from UTEP-CS.
The department employs one office secretary, who assists the faculties in providing logistics management support related to the project. The information technology team from the CS department at UTEP will help the students with necessary technical support pertinent to work environment and research needs. This includes availing necessary tools, supported software, desktop environments and peripherals, network connectivity, computer password maintenance, and hardware and software configurations.

APPENDIX 3: EQUIPMENT

Computing Facility @ UTEP.
Following is the list of equipment available at PIs' labs at UTEP. A subset of these computers and devices can be used by the graduate assistants to work on various research tasks.

- (4) Dell G5 Tower Workstations
- (6) Dell PowerEdge 740 servers
- (4) Dell PowerEdge T440 Tower
- (4) Dell Mobile Precision 7530
- (2) Alienware m15 Ryzen Ed R5
- (19) Raspberry PIs, 3B+ and 32GB version
- (1) Openflow Software-defined network switch
- (2) Wago PLC Controllers
- (2) 32-set of sensor kit for Raspberry PIs

Since the PIs are part of University of Texas system, they have remote access to the high-performance computing facility availed at **Texas Advanced Computing Center (TACC)**. This will be used rigorously in exercising large scale experiments including coupled machine learning/simulation workflows.

PNNL Research Computing.
Research Computing (RC) is a laboratory-level investment which currently operates two supercomputers.

Constance, is a 520-node computing cluster and features a dual-socket Intel Haswell E5-2670v3 (12-core-per-socket, running at 2.3 GHz) with 64 GB of 2133 MHz ECC memory, a Fourteen Data Rate (FDR) InfiniBand network card, and 480 GB local solid-state drive disk storage. Constance is housed within CSF and is the successor to the original PIC base capability, Olympus. In addition, Institutional Computing supports 5 PB of storage, an increase in long-term storage for projects from the previous 3.5 PB available.

Deception, is a 96-node cluster boasts a total of 6,144 CPU cores powered by AMD EPYC 7502 CPUs. The system has been designed to be easily scalable up to 330 compute nodes and 20,000 compute cores. Deception also supports our Machine Learning (ML) efforts and is equipped over 50 compute nodes with a variety of Nvidia GPUs. In total, over 200 GPUs are available ranging from the P100 architecture through the latest A100 GPUs equipped with 80GB of GPU memory.

Projects use Research Computing resources through a model where the project is charged a competitive rate for cycles and storage that are used. These service offerings include HPC time, Lustre/BeeGFS/NFS/Archive storage, GPU time on Deception nodes, and our internal OpenStack Cloud.

OpenStack Institutional Research Cloud. Through RC, PNNL offers researches access to an OpenStack Cloud infrastructure comprised of 50 Intel based nodes connected together with 40 Gb/s networking. The Research Cloud provides staff the opportunity to explore Cloud technologies

Clusters. PNNL has approximately 20 clusters, ranging from 8-node special purpose clusters to Constance, the 500-node cluster located at the CSF. The clusters on PNNL's campus have a variety of AMD and Intel processors, as well as several different interconnects, ranging from Gigabit Ethernet to HDR-200 InfiniBand.

Junction Cluster. The 48-node Junction cluster is equipped with AMD EPYC CPUs, Xilinx SN1000 SmartNICs, Xilinx VCK5000 FPGA, and AMD MI100 GPUs. This exploratory system was commissioned in 2021 to investigate heterogeneous computing models.

Saudade. Saudade is a half rack system containing 8 SambaNova Systems Reconfigurable Dataflow Unit™ (RDU)s. SambaNova Systems Reconfigurable Dataflow Unit (RDU) is the industry's next-generation processor and is at the core of SambaNova DataScale. RDUs are designed to allow the data to flow through the processor in ways in which the model was intended to run, freely and without any bottlenecks. RDUs eliminate constant data caching and excess data movement inherent to today's core-based architectures. This unlocks significant silicon utilization to unleash more compute than any other solution available today.

APPENDIX 4: DATA MANAGEMENT PLAN

1. **How data and software generated in the course of the proposed research will be shared and preserved:**

   (a) Results from modeling and analyzing the MLPerf HPC benchmarks will be contributed to the MLPerf HPC working group to be shared and preserved in an MLCommons GitHub repository at https://github.com/mlcommons. MLCommons uses the Apache 2.0 open source license.

   (b) The performance models developed by this project, along with documentation on how to use them, will be shared and preserved in a public GitHub repository with the BSD open source license.

   (c) Extensions developed for Score-P will be contributed to the Score-P code base which is publicly available and maintained and preserved by the Virtual Institute - High Productivity Supercomputing at https://www.vi-hps.org/projects/score-p/ . VI-HPS uses the BSD open source license.

   (d) The framework for instrumentation and analysis of coupled machine learning/simulation workflows, along with documentation on how to use it, will be shared and preserved in a public GitHub repository with the BSD open source license.

2. **Plan for making all research data displayed in publications resulting from the proposed research open, machine-readable, and digitally accessible to the public at the time of publication:**
   For publications in journals, data and software used to produce graphs, charts, images, etc., will be made available as supplementary information along with instructions on how to reproduce the displayed results. If the journal does not provide for supplementary information, a link will be provided in the publication to a publicly available repository containing the information. For publications in conferences that provide for artifact evaluation, the guidelines for artifact evaluation will be followed. If the conference does not provide for artifact evaluation, a link will be provided in the publication to a publicly available repository containing the information.

3. **Consult and reference available information about data management resources to be used in the course of the proposed research:** When we use resources at NERSC, OLCF, ALCF, or PNNL, we will follow the guidelines at the facility for management of data produced at the facility. Because the links at https://science.osti.gov/Funding-Opportunities/Digital-Data-Management/Resources-at-SC-User-Facilities to data management resources at the ASCR facilities appear to be broken, the information could not be consulted.

4. **Protect confidentiality, personal privacy, Personally Identifiable Information, and U.S. national, homeland, and economic security; recognize proprietary interests, business confidential information, and intellectual property rights; avoid significant negative impact on innovation, and U.S. competitiveness; and otherwise be consistent with all applicable laws, and regulations:** The project will not produce any data with Personally Identifiable Information. Project personnel will follows all rules and regulations for proprietary interests, business confidential information, and intellectual property right for any resources used.

APPENDIX 5: PROMOTING INCLUSIVE AND EQUITABLE RESEARCH (PIER) PLAN

The University of Texas at El Paso (UTEP) and the Computer Science (CS) department greatly value diversity, equity, and inclusion (DEI). UTEP incorporates inclusive excellence in its operations and efforts to provide access to an exceptional education for all students. The Edge, UTEP's student success framework, provides students with access to diverse, challenging, and enriching practices, pairing opportunities with students' strengths and aspirations. The CS department leads the NSF-funded INCLUDES Computing Alliance of Hispanic-Serving Institutions (CAHSI), a national network of Hispanic-Serving Institutions and organizations from the public and private sectors committed to Hispanic and female student recruitment, retention, and advancement in computing. UTEP was ranked first in the nation by a Brookings Institution study for its success in achieving both competitive research and student social mobility, and is among the top ten U.S. universities for helping graduates move from family incomes in the bottom 20% to the top 20% according to Washington Monthly.

The above institutional and department commitment to DEI provide resources that our research project can draw on to provide research opportunities for students from underrepresented groups in scientific computing and machine learning. One example is the CAHSI-Google Dissertation Award. CAHSI works with Google and the CMD-IT Diversifying Future Leadership in the Professoriate Alliance (FLIP) to increase the diversity of PhD students in computing. They invite doctoral students from traditionally underrepresented backgrounds to apply for the awards to be used for the last year of the completion of the dissertation requirements. Such an award could help connect our project with the CAHSI community and provide funding for PhD dissertation completion on a research topic related to the project.

This project will leverage an institutional PNNL-UTEP partnership that seeks to recruit UTEP students for research internships and involve UTEP faculty and students in research with PNNL scientists [19]. This partnership aims to increase internships and employment, facilitate joint appointments, promote board/committee memberships, explore research investments, and submit joint proposals. These efforts will help establish a diverse and inclusive pipeline for advancing students from underrepresented groups toward HPC/AI careers including at DOE laboratories. By partnering with existing efforts, we can better steward DOE investments in tackling the DEI challenges in HPC.

The specific activities that will be carried out by this project to promote DEI include the following:

- Collaboration with other CS faculty involved in research and teaching in machine learning and AI to work on joint research together with their students. Such collaboration will help achieve critical mass to establish a group of students at different levels working on machine learning topics, with more advanced students mentoring and serving as role models for less experienced students.
- Integration of scientific computing and scientific machine learning topics into existing parallel computing, computer networking, and machine learning classes in order to encourage undergraduate students to consider research careers in these fields.
- Presentations and discussion by our PhD students about their research in department seminars, international conferences, and meetings of student organizations such as ACM-W.
- As part of this project, PNNL personnel will offer seminars, brown bag discussions, workshops, and onsite visits to increase awareness of research and career opportunities.
- Showcase research outcomes of graduate students through the nationally recognized Great Minds in STEM (GMiS) conference where students will present their research posters, and organize hackathons in the realm of scientific computing and machine learning.

APPENDIX 6: OTHER ATTACHMENTS

1 Pacific Northwest National Laboratory Letter of Support

902 Battelle Boulevard
P.O. Box 999, MSIN J4-30
Richland, WA 99352
(509) 372-6791
joshua.suetterlein@pnnl.gov
www.pnnl.gov

January 30, 2023

Dr. Shirley Moore, Associate Professor
University of Texas at El Paso
Department of Computer Science
500 West University Avenue
El Paso, Texas 79968

Subject: Collaboration for DOE Office of Science FY2023 Funding for Accelerated, Inclusive Research (FAIR); DE-FOA-0002931

Dear Dr. Moore:

If your proposal entitled, "Performance and Scalability of Distributed Deep Learning," is selected for funding under the SC FAIR FOA, it is my intent to collaborate in this research by providing domain expertise and state-of-the-art facilities to develop performance models for scalable deep learning pipelines.

Thank you for the opportunity to participate.

Sincerely,

Joshua Suetterlein, Ph.D.
Computer Scientist
High Performance Computing Group
Physical and Computational Sciences Directorate
Pacific Northwest National Laboratory