Project Summary

Overview:

Large Language Models (LLMs) have gained significant attention in recent years for revolutionizing the field of natural language processing (NLP) for language understanding, knowledge acquisition and generating complex human-like reasoning in various application domains such as virtual assistants, chatbots, and healthcare. In an effort to make these models more powerful, recently, the monolithic LLMs have exploded in model size and number of parameters, thereby becoming almost prohibitive in terms of compute resource and memory requirements as well as training costs. This has resulted in an impediment for academic institutions and smaller enterprises to make significant contributions to the advancements of LLMs. Therefore, it is critical to take a fresh look at LLM design for providing a democratic platform for contributing to cost-effective innovations in LLMs. One promising direction that has been recently explored is in developing modular architectures, such as Mixture of Experts (MoE) and Composition of Experts (CoE) that enable flexibility, scalability, and resource efficiency. However, this paradigm is in its infancy and the current solutions represent specific design points in a vast search space. This proposed research plans an ambitious cross-layer (algorithms/models, systems and architecture) investigation to advance such modular LLM models by exploring a novel Ensemble of Experts (EoE) framework for designing scalable, application-tailored and adaptable LLM models for improving training and inference efficiency in terms performance, energy efficiency, carbon footprint, and fault-tolerance.

Keywords: Chiplet-Based Systems, Large Language Models, Ensemble of Experts, Hardware-Software Co-design, Algorithmic and Scheduling Support.

Intellectual Merit:

In this context, the proposed EoE framework is envisioned to be a network of expert language models, each trained on domain-specific datasets for providing scalability, modularity, and inherent fault-tolerance. The proposed research consists of 4 intertwined thrusts. Thrust-1 is aimed at investigating the algorithmic foundations for the design and dynamic plug-and-play configuration of EoE for application-specific training and inferences. For executing these morphable experts on an underlying hardware, Thrust-2 focuses on investigating system-level support such as the scheduling of experts considering data locality and memory hierarchy management. Thrust-3 examines architectural design choices for efficient mapping of experts to hardware, leading to a chiplet-based design consisting of compute engines such as CPUs, GPUs, and accelerators. This thrust also investigates the required mechanisms for minimizing data transfer overheads, and the underlying interconnect architecture to facilitate reconfigurability and fault tolerance. Finally, Thrust-4 is devoted to developing a comprehensive empirical evaluation platform consisting of a simulator and analytical tools to evaluate and validate our design trajectory in terms of performance, energy efficiency, and model accuracy.

Broader Impacts:

Considering the importance of designing cost-effective LLM models for various application domains, if successful, the project will lead to a more systematic, scalable, robust, customized and cost-effective LLM models for various application domains. The proposed scalable cross-layer framework will enable exploration of novel architectural and system-level solutions in addressing the LLM design challenges. On the educational front, our plans include development of two new courses, and involvement of undergraduate and graduate students in this research, where they will get exposure to cross-cutting topics in LLM algorithms/models, systems and computer architecture. The PIs will recruit female and minority students to work on the project. Additionally, industry collaboration for realistic design decisions and potential technology transfer will be pursued. The tools developed through this project will be disseminated in the public domain to other researchers and practitioners. The PIs will undertake several Broadening Participation in Computing (BPC) activities aligned with the departmental BPC plan such as summer camps for girls, and work in collaboration with the Penn State College of Education to expose K-12 students to many areas of computer science in general and the emerging LLMs in particular.

Project Description

1 Introduction

The advent of transformer-based architectures [157] has revolutionized the field of natural language processing (NLP), exhibiting unprecedented capabilities in understanding the nuances of human language, maintaining rich context, and generating human-like responses. At the forefront of these transformative advancements are Large Language Models (LLMs), which have led to a paradigm shift in NLP and artificial intelligence (AI) at large. Their seamless integration into everyday life across diverse domains—such as virtual assistants [12, 16, 51], customer service chatbots [6, 121], code generation tools like GitHub Copilot [48], and knowledge management systems like NotebookLM [52]—has profoundly impacted how we interact with technology. According to the 2023 McKinsey Global Survey on AI [4], 65% of respondents indicated that their organizations are now utilizing generative AI technologies powered by LLMs, a figure that has nearly doubled from a survey conducted ten months earlier. In healthcare for instance, more than 40% of institutions are leveraging LLMs to enhance patient care through improvements in diagnostics, patient support, and documentation efficiency [122, 130, 142, 176]. LLMs have also made significant inroads into fields traditionally dominated by human creativity, such as creative writing and AI-generated artworks [3, 45, 85, 125, 177]. Furthermore, in the retail and business sectors, LLM-powered AI chatbots have been shown to reduce the time taken to process an order by 50% to 70%, demonstrating significant efficiency gains even in industries traditionally reliant on human interaction.

What are the Problems? While LLMs have become integral across various sectors due to their advanced capabilities, constructing these models from scratch for specialized applications presents significant challenges due to the prohibitive cost. For instance, the Megatron-Turing 530B model was trained using 2K A100 GPUs over a duration of 3 months consuming over 3 million GPU hours [149]. Similarly, it took 384 A100 GPUs to train BLOOM over 3.5 months [165] and 6144 TPU v4 chips were used to train PaLM-540B model over 50 days [27]. The elaborate resources and the extensive times that these training tasks entail are indicative of severe financial implications of running these models. In fact, a recent study from CSET [100] estimates that the cost of building LLMs will move to trillions in roughly 36 months!

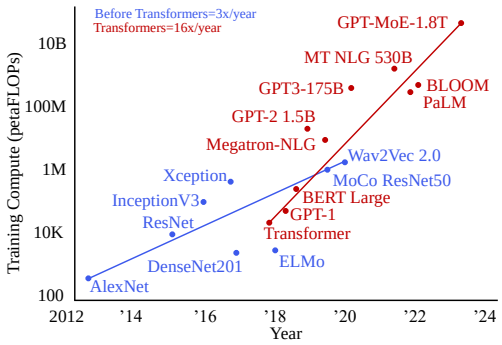

Figure 1: *Trend of increasing compute power requirements for training. Source: [30].*

These numbers are astonishing and underline the intense interest and investment that this domain has garnered. It is also equally important to note the impact of LLMs on the environment. An LLM like PaLM-540B to be trained in a data center facility operated on 89% carbon-free energy still produces 271.43 tCO2e, which is compared to be similar to emissions of a direct round trip of a single passenger jet between San Francisco and NYC [27]. A critical but often overlooked aspect of these compute-intensive operations are usage of water in cooling mechanisms in the data centers. A study [94] on the same highlights the operational water consumption footprint for LLM training and inference and points out that, in the United States, on average, for every 30 inference requests on a small model like GPT-3 results in consumption of 500mL of water. In fact, for training GPT-3, on an average, 5.4 million liters of water is consumed!

Clearly, such extensive requirements create barriers for academic institutions and smaller enterprises for advancing the state-of-the-art in LLM cost-effective training, inference, and adaptability. In addition, the availability of clean and high-quality data is reaching physical limits. As models grow larger, they require exponentially more data, but the Chinchilla Law [64] indicates that, beyond a certain point, increasing model size without proportional data scaling yields diminishing returns. This scarcity of high-quality data constrains the effective training of larger models. Furthermore, as depicted in Figure 1, Bill Dally, the Chief Scientist at NVIDIA, summarizes [30] how the compute needs for training have increased with model complexity, placing immense strain on existing computing systems. The resulting power needs have recently triggered data center providers to install their own power plants [44, 53, 136, 155]. Current hardware accelerators like GPUs, and even specialized processors like Groq [61], Cerebras [86, 96], Graphcore [54, 115], and SambaNova [131], are not efficiently utilized due to limitations in memory bandwidth, interconnects, and data handling capabilities. These hardware inefficiencies lead to suboptimal performance and increased energy consumption. Finally, the inherent limitations of monolithic models used in most of the current

LLM applications [12, 16, 27, 51, 106, 123, 124, 149]— such as lack of domain specificity and inflexibility in continual learning—slow down the scientific discovery cycle. These factors collectively restrict innovation, limit accessibility, and contribute to significant environmental impact due to high power consumption and carbon emissions, conflicting with global commitments to achieve carbon neutrality [14, 17].

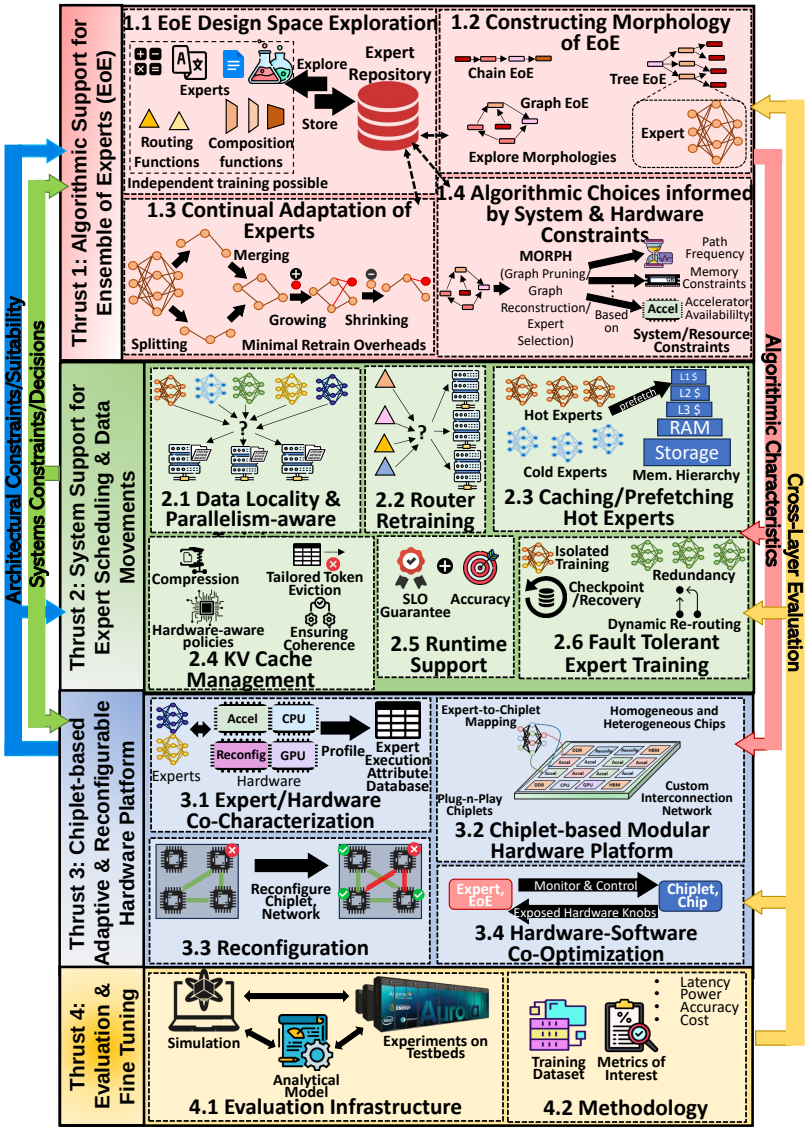

Figure 2*: Overview of the proposed project. The included numbers indicate task IDs. See our timeline (Figure 8) for time-span of the individual tasks.*

What is the Solution Space? Addressing these challenges necessitates a fresh look beyond the current monolithic design for providing a democratic platform for contributing to cost-effective innovations in LLMs. One promising direction that is being recently explored is the development of "modular" architectures that enable flexibility, scalability, and efficiency. Mixture of Experts (MoE) [145] and Composition of Experts (CoE) [131, 151] models have been proposed to distribute computational loads across multiple specialized networks. The new CoE architectures, which use a combination of small monolithic or MoE models, introduce new opportunities by allowing experts to be trained independently and incrementally, thereby reducing computational resource requirements, while enhancing fault-tolerance, and enabling the use of custom accelerators. However, existing MoE and CoE models have significant limitations. The MoEs themselves are still monolithic models albeit having intra-model sparsity, and parameter-bloating [134] in MoE leads to large training resources. Furthermore, the implications of training these MoEs in terms of required architectural and system support, overall training time, accuracy, and dynamic adaption in different application domains have not been systematically investigated. Additionally, there exists significant load imbalance among experts, increased communication overhead, and complexity in routing mechanisms [42,74,131,151]. Thus, while the MoE/CoE design paradigm is promising, the overall solution space is little explored, especially for complex compositions of smaller experts. Therefore, it is imperative to investigate the modular LLM design space in depth not only to mitigate the above issues, but also to facilitate democratization by allowing anyone to use and contribute in a "plug-and-play" fashion. These observations call for a holistic hardware-software "co-design" that integrates efficient expert models with custom system support and reconfigurable hardware architectures, to optimize performance, while minimizing resource consumption.

How do we get there? Our proposal introduces an **Ensemble of Experts (EoE)** framework that embodies this co-design philosophy. We envision a network of expert language models, each trained on domain-

specific datasets, forming a modular and scalable system. This approach defines a large set of different expert types, routers and composition functions that can be used to build an "ensemble" (model) that is customized for the application at hand, and allows for independent training and updating of experts, facilitating continual learning and adaptability. By enabling plug-and-play functionality, our design should empower users to customize and extend the model by adding new experts, dropping existing experts and changing the connections among experts, routers and composition functions, fostering collaboration and innovation within the community. To support this modularity, we also propose a flexible system architecture and runtime that efficiently handles queries. Smart routers dynamically direct queries to the most relevant experts based on domain relevance and operational dynamics, optimizing performance and resource utilization. This dynamic routing reduces computational overhead and energy consumption, helping in addressing the power efficiency concerns. Complementing the modular models and systems, we advocate for the use of "chiplet-based" hardware accelerators. These hardware components are designed to be flexible and scalable, allowing customization to specific computational needs. By leveraging reconfigurable chiplet-based architectures, we aim to optimize hardware efficiency, reduce power consumption, and minimize the environmental footprint of AI operations. Through this integrated approach, we aim to lower the barriers to entry for AI development.

In this context, we propose a research plan consisting of 4 intertwined thrusts. Thrust-1 is aimed at investigating the algorithmic foundations of our EoE paradigm that consists of an ensemble of experts, which will facilitate application-specific morphable LLMs. For executing these morphable experts on an underlying hardware, Thrust-2 focuses on investigating system-level issues in EoE training and inference, focusing in particular on expert scheduling and memory hierarchy management. Thrust-3 examines architectural design space for efficient mapping of experts to hardware leading to a chiplet-based design consisting of a heterogeneous platform of compute engines such as CPUs, GPUs, and accelerators. This thrust will also investigate the required mechanisms for minimizing data transfer overheads and the underlying interconnect architecture to facilitate reconfigurability and fault-tolerance. Finally, Thrust-4 is devoted to developing a comprehensive empirical evaluation platform consisting of a simulator and analytical tools to evaluate and validate our design in terms of performance, energy efficiency, and model accuracy. The concept of an "ensemble" serves as a foundational element, uniting our diverse research thrusts: a comprehensive set of models, a suite of accelerators, and an array of simulation and evaluation frameworks. Our modular, plug-and-play based approach not only allows for targeted research within individual elements of the ensemble, but also supports contributions from the broader research community. Such incremental growth epitomizes the democratization of large-scale design efforts. Thus, we believe our proposal is ambitious as its potential to revolutionize the training and deployment of LLMs is profound.

2 Proposed Research

A high-level view of our proposed research consisting of 4 thrusts is depicted in Figure 2. In addition to the individual tasks in each thrust, Figure 2 also shows the inter-task dependencies as well as interactions between them, highlighting our cross-layer co-design aspect.

2.1 Thrust-1: Algorithmic Support for Ensemble of Experts

The main focus of this thrust is to investigate different models and algorithms for Ensemble-of-Experts (EoE) systems, paying special attention to search space optimization for "morphable" LLM expert ecosystems, "dynamic" expert networks, and obtaining new experts from existing ones. Our research tasks in Thrust 1 are illustrated in Figure 3. They will address the following research questions: i) *What is the search space for EoE systems, in terms of expert types, expert routing, expert model architecture, and expert composition?*; ii) *How can we dynamically shape the morphology of EoE systems by identifying the best design points in the search space?*; iii) *How can we continually adapt experts through expert merging, splitting, growing, and shrinking?*; and iv) *How do compute and memory resource constraints and runtime environments influence our algorithmic decisions?* This thrust aims to advance the state-of-the-art by proposing an innovative framework built on a network of specialized LLM experts that can operate both independently and in collaboration. The modular nature of our system allows experts to be trained separately and then combined in multiple configurations. This offers two key advantages: first, it enables agile adaptation and targeted training to respond to evolving knowledge domains and user requirements; and second, it allows for optimization across different system layers to improve both architectural efficiency and overall performance.

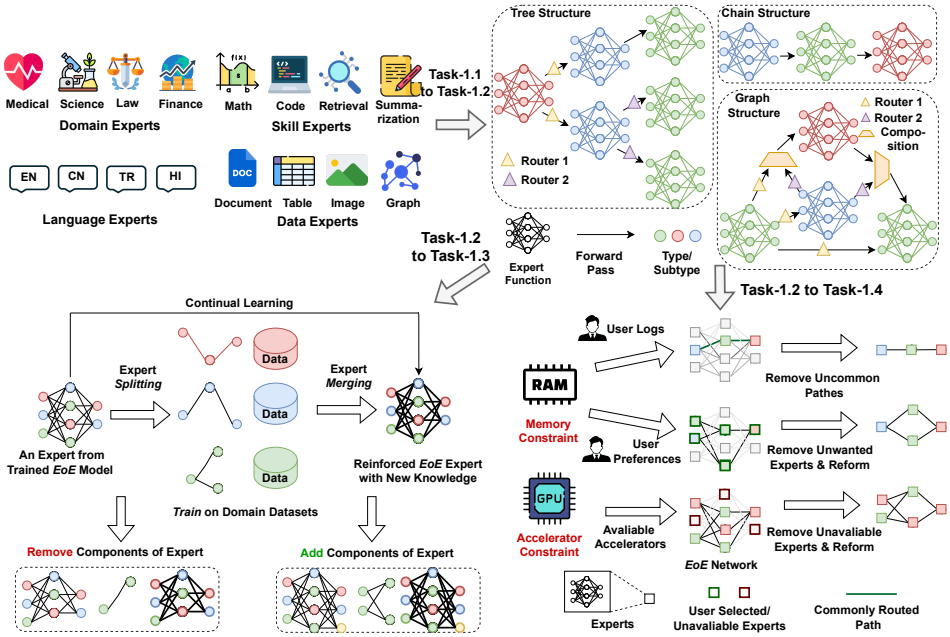

Figure 3*: An overview of four tasks in Thrust-1. Task-1.1: EoE Design Space Exploration. Task-1.2: Constructing Morphology of Ensemble of Experts. Task-1.3: Continual Adaptation of EoE through Morphable LLM Experts. Task-1.4: Algorithmic Choices Informed by System and Hardware Constraints.*

Task-1.1: EoE Design Space Exploration

In order to address the limitations of the state-of-the-art monolithic LLMs, in this task, we propose to make a paradigm shift by laying the algorithmic foundations for an EoE system, a novel LLM framework consisting of numerous expert models that can be composed into an ecosystem *dynamically* according to the user query. The main objective of this morphable design is to minimize the resource requirements and training/inference time compared to gigantic monolithic models, while maintaining required accuracy needs. Each expert model is a smaller language model with several orders of magnitude fewer parameters versus monolithic LLMs such as GPT-4 [124]. Here, each expert is specialized to solve tasks in specific domains, languages, or skills. Experts can be trained individually and continually adapted to generate new experts. Expert models are also "composable", so that they can be chosen dynamically according to inputs and then grouped together to solve difficult tasks, which require a combination of skills and knowledge sources. Towards this, we aim to explore a vast design space of model architecture choices and expert model types.

A Unified View of EoE Function Choices. We define three building blocks of an EoE system: "Routing", "Expert", and "Composition". Our exploration will focus on various types of router functions to direct user queries to appropriate experts, different expert model architectures, and composition functions to *aggregate* expert outputs. These components form the foundation for constructing a "morphable ecosystem of experts" through dynamic routing and composition. The different methods for each component are summarized in Table 1. Each expert can vary in model size and architecture depending on the complexity of its domain. Note that *existing works have only studied a few specific points in this vast design space*, lacking a comprehensive exploration of this search space of expert model design and implementation. For example, recent research on mixture-of-experts (MoE) mostly focused on implementing experts as sparse feed-forward neural network (FFNN) layers in an end-to-end transformer [74], and thus they still need to be trained as a single model, and not flexible to be trained or fine-tuned separately. The other line of research [131, 151] implemented experts as individual language models, yet the experts are selected to work individually and they cannot be composed dynamically to complete complex tasks. Our approach will provide a unified view of these functions and analyze a diverse combination of routing, composition, and expert functions to improve the robustness and efficiency. We unify these three types of functions in two steps.

First, we decompose EoE networks into three parts, including routing, expert and aggregation functions, constructing a repository of function choices for each type. Next, we evaluate the new combinations by extracting one function from each type. In this way, we aim to find the optimal combination of the choices.

Expert Repository. For expert types, as shown in Figure 3, we will cover different domains such as scientific, medical, legal, and education, skills such as math reasoning, coding, question answering, and summarization, data modality such as structured data (tables, databases), unstructured data (documents, speech transcripts), semi-structured data (XML, JSON), and images, and languages such as English, Chinese, Turkish, and many other low-source languages. Under each type, there can be subtypes of experts. For example, there can be different subtypes of science experts for different scientific disciplines. We will maintain a repository of LLM experts that are independently trained, from which we can select and modify to form an ecosystem of LLM experts for complex tasks in continually evolving usage scenarios. This repository can constantly expand in types and sizes by welcoming open-sourced, community-contributed models. It delivers the promise of scaling LLMs through many smaller, specialized, independently trained expert language models. Our LLM expert repository, together with our cross-layer approach among algorithm, system, and architecture, will lay the foundation for the democratization of LLM development under friendly compute budgets for a wide range of community groups.

	Method	Function
Routing Function	Mixture-of-experts	$\alpha_i = \text{SoftMax}(\mathbf{W}_i\mathbf{h}_{i-1})$
	Variable size	$\alpha_i = \{a > k \vert a \in \text{SoftMax}(\mathbf{W}_i\mathbf{h}_{i-1})\}$
	Top-k learned routing	$\alpha_i = \text{TopK}(\text{SoftMax}(\mathbf{W}_i\mathbf{h}_{i-1}))$
Expert Function	Feedforward network	$\mathbf{h}_i = \underline{\mathbf{W}_i}\mathbf{h}_{i-1}$
	Prompt tuning	$\mathbf{h}_i = \underline{f_{\theta_i}(\phi_i}, \mathbf{h}_{i-1})$
	Low-rank adaptation	$\mathbf{W}_i = \mathbf{W}_i' + \underline{\mathbf{AB}}$
	Large language model	$\underline{\theta} = \text{argmax}_{\underline{\theta}} \prod_{t=1}^{T} p_{\underline{\theta}}(y_t \vert x, y_{j<t})$
	Retrieval augmented generation	$\underline{\theta} = \text{argmax}_{\underline{\theta}} \sum_{z\in\mathcal{D}} p_{\underline{\theta}}(z\vert x) p_{\theta'}(y\vert x, z)$
Composition Function	Representation averaging	$\mathbf{h}_i = \sum_{j=1}^{M} \alpha_i \mathbf{h}_{i,j}$
	Weight summation	$\mathbf{W}_i = \sum_{j=1}^{M} \mathbf{W}_{i,j}$
	Sequential aggregation	$f_{\theta'} = f_{\phi_M}(f_{\phi_{M-1}}(\cdots f_\theta))$

Table 1*: Different methods for three building blocks of our EoE framework: Routing, Expert, and Composition. α indicates the weight of M experts, i is the index of the layer in LLM, $\boldsymbol{W}, \boldsymbol{A}, \boldsymbol{B}$ and $\boldsymbol{h}$ are the weight matrix and hidden vector, f_θ is the neural network component parameterized by θ. x is the input text, $\mathcal{D}$ is the document for retrieval, and T is the length of the generated text. Underline indicates the expert function.*

Task-1.2: Constructing Morphology of Ensemble of Experts

In this task, we propose innovative, flexible, and diverse ways of connecting individual experts to create diverse LLM expert network morphologies, tailored to various usage scenarios by applying different combinations of expert types, routers, models, and composition functions. In Figure 3, we illustrate three specific morphologies: *chain*, *tree*, and *graph*, among other potential configurations. The chain structure connects experts in a sequential manner, ideal for tasks requiring a series of expert skills; the tree structure models a type-subtype hierarchy; and the graph structure enables complex expert communication and collaboration. Each morphology is described in detail below:

Modeling Diverse Expert Types with Chain Ensemble-of-Experts. Experts can be classified via a set of diverse criteria a, such as domain, skills, and languages. We intend to model this structure by building a **C**hain Ensemble-**o**f-**E**xperts (CEoE), where layers are assigned for different classification criteria a, and each expert in each layer is assigned for a type t under its classification a, such as math in skills, medical in domains. During training, a document is routed to its type in criteria a_1 and then routed to its type in another a_2 until all criteria in a are covered. For instance, a Chinese medical summarization document will be routed to the Chinese language, medical domain, and summarization skill experts layer by layer. Our solution will leverage our previous work on multi-stage summarization [186] and multi-agent framework for long-context tasks [187].

Modeling Expert Type-Subtype Hierarchy with Tree Ensemble-of-Experts. Since knowledge domains often exhibit a hierarchical structure, it is natural to model an EoE using a hierarchical approach to save training and inference costs further. Motivated by this, we propose a novel framework **T**ree **E**nsemble-**o**f-**E**xperts (TEoE). Specifically, given a knowledge domain of k layers, we assign the structure to k adjacent FFNN expert layers in an EoE model, where the ith layer indicates the ith level of the knowledge. For in-

stance, in a three-layer knowledge domain, NLP is under Artificial Intelligence (AI), which is under Computer Science (CS). When training, a document of NLP is first routed to CS in the first layer, then routed to AI, and NLP. This procedure repeats $\lfloor L/k \rfloor$ times in a single run where L is the number of layers in LLM.

Modeling Expert Collaboration with Graph Ensemble-of-Experts. By integrating multiple Chain and Tree EoE structures, we can create complex Graph Ensemble-of-Experts (GEoE), facilitating communication and collaboration among experts. For example, each expert in chains and trees can produce intermediate results, and they can be aggregated into the final manager expert to produce the ultimate result.

Task-1.3: Continual Adaptation of EoE through Morphable LLM Experts

LLMs require continual learning to keep pace with rapidly evolving knowledge and user demands. However, traditional monolithic LLMs are not well-suited for frequent updates, because training monolithic LLMs often involves wholesale replacement which is prohibitively expensive and challenging. By contrast, *our proposed EoE paradigm is inherently modular*, requiring updates of only a small subset of experts to accommodate new usage scenarios. This allows us to perform flexible and efficient "neighborhood training" by *localizing* the training parameters and *freezing* the rest of the parameters as much as possible to save cost. To this end, we propose a method for continual adaptation of EoE systems by *LLM expert split, merge, grow, and shrink*. For example, as shown in Figure 3, we can split a large expert model into smaller domain-specific models and train them on smaller, focused datasets in corresponding domains. After training, we merge them in the final stage to form a reinforced model. Each expert can also grow to absorb new knowledge and shrink to discard obsolete knowledge.

Domain-Based Expert Splitting and Merging. We propose a novel and efficient approach for continual learning of EoE. We split a large expert into smaller, domain-specific experts, train them on focused domain-specific datasets, and then merge them back into a reinforced model. This strategy is particularly useful when we need a single model to handle diverse knowledge sources or when we aim to reduce training costs by first training smaller models and then combining them. To create a smaller model for a domain-specific dataset $\mathcal{D}$, we will identify parameters that have the least influence on loss $\mathcal{L}(\mathcal{D})$ and remove them from the large expert. The importance of each weight at index i, denoted as I_{W_i}, can be approximated by: $I_{W_i} = |\mathcal{L}(\mathcal{D}) - \mathcal{L}_{W_i=0}(\mathcal{D})|$, where $\mathcal{L}_{W_i=0}(\mathcal{D})$ is the loss, by setting parameter W_i to zero. Our solution will leverage our previous work on LLM structured pruning [140], unstructured pruning [181], and parameter-efficient fine-tuning [32, 33]. We will explore several directions for estimating I_{W_i} efficiently such as using a memory-efficient zeroth-order optimizer to estimate gradients using only forward passes [101]. To merge experts into a single one, assuming we have n experts each with parameter $\theta_i, i = 1, \ldots, n$, we merge them into one model θ_m by computing a weighted average of parameters where the weight is each parameter's Fisher information: $\theta_m = \sum_{i=1}^{n} F_i\theta_i / \sum_{i=1}^{n} F_i$, where F_i is the Fisher Information for θ_i.

Knowledge Adaptation via Expert Growing and Shrinking. To incorporate new knowledge, we will explore novel algorithms for expert-growing via life-long learning [147]. We will freeze some neurons of the expert while expanding the network with new parameters and retraining it on new datasets. On the other hand, for knowledge integrity and resource efficiency, we will also explore innovative directions for expert shrinking by unlearning outdated knowledge that is no longer required. To maximize the unlearning effectiveness, given a seed forget dataset, we will perform deductive reasoning based on our previous work [62] to generate a larger forget dataset to account for the ripple effect of knowledge update [28]. We will then perform "unlearning" by parameter-efficient fine-tuning to update the most relevant parameters.

Task-1.4: Algorithmic Choices Informed by System and Hardware Constraints

Our EoE system optimization is fundamentally a co-design problem across the algorithm, system, and architecture layers. As the above tasks develop optimal algorithms for dynamic morphable LLMs, they introduce many types of "affinities" (listed in Table 2) to facilitate our system and architectural level optimizations, which are investigated in Thrusts 2 and 3, respectively. Notably, the reverse is also true, and thus, this task will investigate the influence of resource constraints and execution environments on our algorithmic choices. We will conduct the following research to incorporate the availability of expert accelerators, memory constraints, and workload balance to inform our algorithmic choices:

Selecting Experts based on Available Expert Accelerators. Our LLM expert repository offers a rich set of diverse skills and domains to serve a user query. As a user query can be possibly answered by different sets of experts, we will choose experts based on the combination of two factors: (1) The relevance of expert to the user query, and (2) the availability of accelerator (developed in Thrust 3) for the expert. When two

Affinity	Optimization Stage		Tasks		
	Training	Inference	Algorithm	System	Architecture
Expert–Expert	✓	✓	Task 1.2, 1.3, 1.4	Task 2.1, 2.3, 2.5	Task 3.2
Expert–Data	✓		Task 1.1, 1.3	Task 2.1, 2.5	
Expert–Router	✓		Task 1.2, 1.3	Task 2.2, 2.5	Task 3.3
Expert–Composition Function	✓	✓	Task 1.2, 1.3	Task 2.3, 2.5	
Expert–Accelerator	✓	✓	Task 1.4		Task 3.2

Table 2: *Overview of affinities in the proposal. Definitions of different affinities:* ***Expert–Expert:*** *the tendency of certain experts to be used more frequently together to serve a request;* ***Expert–Data:*** *the tendency of similar experts to use a common subset of training data;* ***Expert–Router:*** *the tendency of given experts and routers to be used together frequently to serve different requests and to share a subset of training data.;* ***Expert–Composition Function:*** *the tendency of some experts to use the same composition function frequently across different requests;* ***Expert–Accelerator:*** *the tendency of some experts to utilize the same chiplet or different chiplets in the same chip.*

experts are equally relevant, we will prioritize those with available accelerators to improve performance.

EoE Graph Pruning for Memory Constraint. Given a memory budget that limits the number of experts the system can host, the EoE network must *adapt* its structure to meet user needs. To this end, we can *prune* the network to a smaller number of experts by using three operations: (1) Remove, eliminating an expert from the graph. This can be achieved by deleting its vector in the gating function g; (2) Combine, combining several experts in one EoE layer. This can be achieved by adding several expert functions (e.g., FFNN) and their corresponding gating functions; (3) Quantize, reducing the memory requirement of experts through quantization [34]. Once the user specifies target domains, the algorithm locates the most irrelevant domains in the graph (either branch or leaf) until the number of experts fits within the memory constraints or the relevance reaches a threshold. Then, remove operations are repeatedly used to prune the system. If the number of experts is still larger than needed, a clustering algorithm (e.g., AGNES [139], and topic clustering [160]) can be employed to combine experts with the highest cluster scores, thus reducing the total number of experts.

Automatic EoE Graph Construction for Workload Balance. The EoE graph pruning process, involving removing and merging experts, can sometimes result in imbalanced branches, where one branch contains significantly more experts than another. This imbalance can lead to inefficiency in workload distribution among different parts of computing resources. To solve this issue, we propose to re-construct the expert graph automatically to balance the workload. We first form a binary tree over all selected domains where the experts are the leaf nodes at the finest granularity. We cluster the experts according to their domain similarity so that each parent node is assigned two most similar leaves. Next, the routers are trained on the new topology, and the leaf nodes are used when the query routes to them.

2.2 Thrust-2: System Support for Expert Scheduling and Data Movements

The primary goal of this thrust is to explore novel system support – targeting *both* training and inference – that complements our algorithmic support for EoE in Thrust 1 and architectural support for EoE in Thrust 3. The representative research questions this thrust will strive to answer include: i) *How can an EoE system be trained in a performance- and energy-efficient manner through the maximization of data locality and parallelism?*; ii) *What are the ways of efficiently retraining routers when new experts are added into or removed from the ensemble?*; iii) *How can fault-tolerance be factored into the training process without an unduly increase in execution latency and energy consumption?*; iv) *How can we identify hot experts and cold experts in LLM inference and how can such information be utilized?*; v) *How should available memory space be managed during training and inference?*; vi) *What are the additional complexities and opportunities KV-caches bring in an EoE environment?*; and vii) *What are the various combinations of algorithmic constructs, runtime software configurations, and architectural features, and how can their "affinity" be synergistically leveraged to enhance the efficiency of training and inference processes?*. We plan to address these questions using 6 different tasks. To enhance the performance of EoE , we propose system optimizations that exploit the spatio-temporal locality/affinity of its components – data, experts, routers, composition functions, and hardware. Table 2 shows how our proposed framework will exploit these affinities across (training, inference) and (algorithm, systems, architecture) dimensions.

Task-2.1: Data Locality and Parallelism–Aware LLM Training
Unlike traditional monolithic LLMs, our EoE -based LLM opens up new opportunities to optimize training for both performance and energy efficiency. Each expert in our EoE can be trained independently and faster, significantly reducing the overall training time compared to monolithic models. Additionally, by clustering and scheduling experts that share training datasets, we propose a "locality-aware" training strategy that minimizes the frequency and volume of data transfers between experts, memory and storage– a key factor in improving both performance and energy efficiency. We plan to model this problem using a "bipartite graph" (refer to Figure 4) where one set of nodes represent disjoint datasets (D1, D2, etc.) and the other represent experts (E1, E2, etc.). An edge between a dataset and an expert indicates that the expert is trained on that dataset (i.e., expert-data affinity). Given memory capacity constraints, we frame the problem as identifying sets of experts to train together such that the data reuse among the selected experts, in a training step, is maximized and the combined data and parameter requirements of the set fit within the available memory. For example, in Figure 4(a), assuming for simplicity that memory can hold at most three experts and three datasets, this locality-aware approach would perform training in three steps. As depicted in Figure 4(b), we train E1, E5, and E7 together as they share datasets, followed by E2, E3, and E6, and the remaining experts. This reduces redundant data transfers, lowering latency, and reducing energy consumption. This approach can be further refined by, for instance, adjusting the weights taking into account a) availability/proximity of appropriate compute to the nodes (data should ideally be present on nodes which can leverage well-suited accelerators with relative ease), and b) load balancing across nodes to prevent overburdening specific nodes (thus, avoiding performance interference).

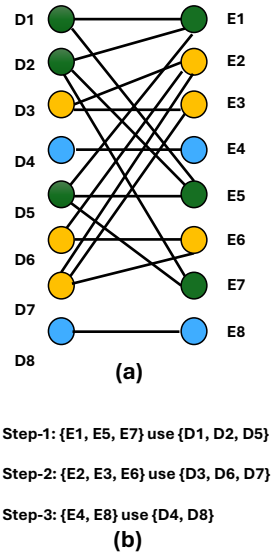

Figure 4: *Bipartite graph to perform data locality-aware expert training. In this case, the training is completed in 3 steps.*

Task-2.2: Router Retraining
As stated earlier, our morphable LLMs will function in a plug-and-play fashion in which we will introduce new experts into the current ensemble incrementally, drop unneeded experts from an ensemble, or replace existing experts with others. All these activities may require *retraining* the routers. We will develop techniques that can help us identify which routers need retraining and when, and schedule such retraining in a data locality and parallelism aware fashion. Since the experts are pivotal in the way the prompt/query is being answered, it is equally imperative for the routers to fully utilize the expert network by directing the relevant queries to the right expert(s) dynamically depending on the current experts being involved in the network. To achieve this, the router needs to be retrained/fine-tuned according to the state of the expert network by taking advantage of expert-router affinity to the greatest extent possible. Unlike conventional networks, where an entire end-to-end training of the experts and routers needs to be done alike, our scheme makes use of a more targeted approach that can significantly reduce the training complexity of the routers. In our methodology, the router only needs to be fine-tuned on the dataset of the expert that it interfaces with, which is much smaller than the entire dataset. As has been highlighted in Table 2, we plan to exploit the expert-router affinity to reduce the training complexity of the router at initial training time and even while fine-tuning as the ensemble of experts evolve. This would ensure that these routers can be fine-tuned in isolation, causing minimum to none ripple effects on to other expert branches, lowering the computational complexity and cost. We intend to co-locate the expert and routers physically in the system during both training and inference time to leverage this affinity and reduce expensive data movement cost.

Task-2.3: Using Hot and Cold Experts: Caching and Prefetching
Clearly, the optimized management of the system memory/storage hierarchy in LLM inference is of critical importance [131, 146]. Towards this, we plan to experiment with various latency reducing/hiding techniques such as caching and prefetching by intelligently using experts based on their frequency of invocation. We will dynamically classify experts as "hot" and "cold" experts based on their frequency of use. Hot experts are those that are more frequently invoked for generating responses because of common or recurring query topics, whereas cold experts are the ones which are less frequently utilized. Note that this categorization is *dynamic*, adapting to the temporal nature of the incoming requests which can have

variations based on user demand. As shown in Figure 5, one way of utilizing this hot/cold expert separation is to store their respective model parameters at different levels in the memory hierarchy, dictated by the degree of hotness, so as to optimize the response generation pipeline. Specifically, the various experts are typically stored on SSDs or HDDs, and are on-demand loaded into the main memory as required by incoming queries. This dynamic allocation allows for a flexible working set of experts hosted in the main memory, tailored to the specific needs of the batch of requests currently being processed. When a new batch of requests is introduced into the system, an initial selection of the top-k experts is conducted for each request in the initial layer of processing. By loading these selected potentially useful and relevant experts into the main memory, we propose to optimize the response time and computational efficiency.

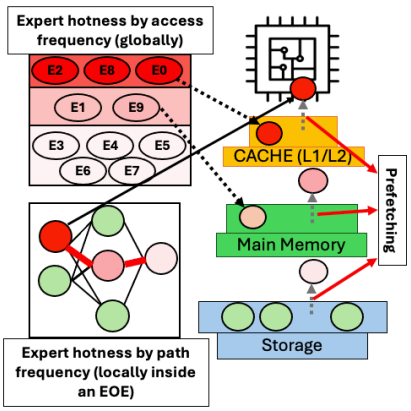

Figure 5*: Hot and cold experts.*

Since this method also reduces the data movements in the system, we can expect a significant reduction in energy consumption. Additionally, we will identify hot and cold "paths" and use this information in *prefetching* the experts and routers (which are actually specialized experts) into the higher levels of the memory hierarchy, close to compute units (note that this is an example use of expert-expert and expert-router affinities). Specifically, leveraging the observed patterns of expert reuse ("hotness" patterns) at different locations in the EoE , our system can intelligently predict and load the experts likely needed in subsequent layers to the main memory. While response is being generated for some prior expert, this predictive loading can take place concurrently for the experts ahead in the path to minimize resource idleness and ensure seamless transition for subsequent processing.

Task-2.4: KV Cache Management

Key-Value (KV) caches in LLMs store past activations to *accelerate* inference by avoiding redundant computations [7, 35, 43, 73, 99, 146, 148, 163, 167, 168, 171, 172, 188, 189, 193]. In large models with billions of parameters, and close to a million context length [27, 124, 149], the KV cache size could grow to hundreds of gigabytes needing careful management. In our EoE , two unique KV cache challenges arise: i) Loading experts and their KV caches onto GPUs for a user can cause high latency and memory consumption due to the initial "prefill" stage, leading to a *cold start* problem. To address this, we plan to prefetch experts *and* their KV caches (leveraging hot and cold experts) or GPU load balancing to select devices with sufficient memory; and ii) Each expert has specific memory requirements for model weights and past KV caches, and must reserve space for future KV caches to prevent out-of-memory errors, which depends on the expert's response length—from single words to thousands of tokens. We plan to address this in two ways. First, we will implement different KV cache management policies for different experts, depending on their accuracy requirements, context lengths, and latency tolerances. And second, we will explore EoE-specific KV cache compression strategies to optimize memory usage including quantization and sparsity optimization techniques. An example would be: first, dividing the available KV cache space among experts and then, across the attention heads in each expert, with the goal of allocating more cache space to experts and attention heads that can benefit most from that space (compared to other experts/attention heads).

Task-2.5: Runtime Support

Our runtime support system will serve as the cohesive glue that efficiently manages system resources, models, and data in real-time with minimal overhead. It effectively addresses the operational efficiency and response quality within the constraints of stringent SLOs and a minimum accuracy rate. By integrating expert affinities (Expert–Expert, Expert–Data, Expert–Router, and Expert–Composition Function) detailed in Table 2, the system will optimize responsiveness and computational efficiency. Grouping experts in close memory proximity is expected to enhance cache efficiency and reduce latency. Data locality should benefit from prefetching and caching critical data, which would lower transfer delays and potentially speed up training and inference. Intelligent routing algorithms are proposed to efficiently distribute queries based on real-time performance data and expert availability. Custom composition functions are anticipated to further speed up the integration of diverse expert outputs, thereby improving overall response times. This unified strategy ensures rapid, accurate query responses and high throughput within the infrastructure's memory and computational constraints.

Task-2.6: Fault-Tolerant Expert Training
Fault-tolerance of monolithic LLM models has been a major concern due to the long-running training jobs on a large number of GPUs [27, 149, 165]. The proposed EoE model can provide inherent fault-tolerance since each expert can be trained independently, and is thus exposed to hardware failures in a reduced timeframe. To further minimize the impact of failures, we will investigate known fault-tolerance techniques from the distributed computing domain to support graceful degradation of training. A few techniques we plan to study include: i) *Exploiting Redundancy*: Selectively training multiple experts on similar/overlapping tasks (which are deemed critical to performance/accuracy) provides a fallback mechanism in case one of them fails. Here, the redundant experts should ideally be distributed across multiple nodes; ii) *Isolated Training*: In the event of changes to experts, our approach enables retraining by accounting for minimum number of neighboring experts/routers; this will inherently facilitate fault tolerance, thereby allowing retraining of the EoE with minimum overhead in the event of expert failures; iii) *Checkpointing/Rollback Recovery*: Techniques such as checkpointing weights/gradients periodically with rollback recovery in the event of expert failures can allow training to progress from the latest checkpoint (versus restarting the entire training); and finally, iv) *Dynamic Expert Re-routing:* In the event that an expert on one node fails, our system-level framework will communicate with the algorithm-level routers to dynamically decide which experts (on which node) should be used instead. Note that such system level fault-tolerant techniques will be augmented with hardware (chiplet)-level fault-tolerant techniques, described later in Task 3.4.

2.3 Thrust-3: A Chiplet-based Adaptive and Reconfigurable Hardware Platform

Our proposed EoE-based LLM algorithm consists of a diverse set of building blocks with routers, experts, and composition functions, entailing heterogeneity in different stages of the application execution. For example, the experts in the input layer processing the input prompt are expected to require a larger KV-cache against experts in the middle or output layers processing intermediate and refined embeddings. Moreover, the experts could differ in their size and thus in their compute and memory requirements. In addition, for supporting morphable network structures that change over time with continual learning, the underlying interconnect should be dynamically reconfigurable. In terms of cost, utilization, power, and area requirements, we expect several inefficiencies with existing off-the-shelf hardware like CPUs and GPUs when executing our models for training, inference, and re-training purposes, thus exacerbating the gap towards democratization. Towards this, in this thrust, we propose to explore a novel "chiplet-based" custom hardware platform. Chiplet-based designs have shown great promise for integrating a variety of modular chips, both homogeneous and heterogeneous, on a silicon interposer [72, 152] and is a great fit for our envisioned modular and fault-tolerant design. The specific questions we would like to address in this thrust comprising of 4 tasks include: i) *Which parts of the EoE network are most preferable for custom hardware acceleration from the performance, power, and accuracy standpoints?;* ii) *What types of chiplet-based architectures are suitable for EoE-based LLMs, and what is the search space for chiplets?;* iii) *What kind of reconfigurability is needed for chips to accommodate the heterogeneous and morphable aspects of EoEs?;* iv) *How can inter-chiplet and inter-chip data movements be choreographed to maximize performance and minimize energy consumption for training, inference, and re-training?*; and v) *What are the hardware-software co-design opportunities towards an efficient cross-stack system?*.

The core of our architectural investigation is "expert–accelerator affinity" (Table 2), which involves dynamically mapping "experts" to the most suitable accelerators with configurations such as *one-to-one*, *one-to-many*, *many-to-one* and *many-to-many*, each providing unique benefits for scalability and efficiency.

Task-3.1: Expert/Hardware Co-Characterization
To efficiently serve various application needs, the question we ask is *what are the intelligent ways to execute EoEs?* As discussed in Thrust-1, our expert repository allows us to plug-and-play experts to create diverse EoEs, thereby many types of experts exist within and across EoEs. With the target of achieving an efficient execution, we first want to understand the execution behavior. Towards this, for each EoE dataflow over varieties of off-the-shelf hardware platforms including general purpose cores, GPUs, and hardware accelerators like TPU [50] and SN40L [131] among others, we will investigate the performance, compute and memory utilization, accuracy contribution, and power metrics for every single expert to list the key bottleneck kernels. Also, we will identify the most common or frequently accessed experts within and across EoEs. For these metrics, the expert execution may favor different hardware devices. Thus, to cater to the execution diversity in experts, we intend to take advantage of "heterogeneous" chiplets. Using the above profiling, for each expert in each EoE when run over a variety of hardware platforms, we can systemati-

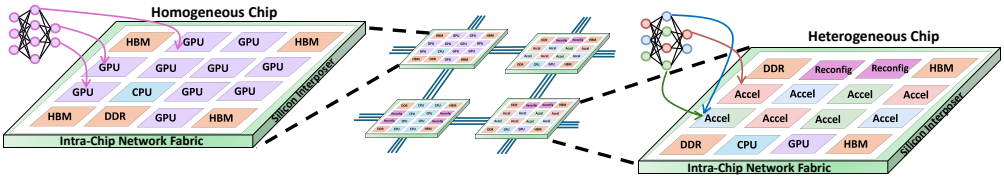

Figure 6: *Our proposed chip infrastructure. Each chip can potentially contain a set of homogeneous or heterogeneous chiplets including accelerator engines, reconfigurable engines, CPUs, GPUs, and various types of memory modules.*

cally create an **Expert Execution Attribute Database** containing metrics such as latency, compute utilization, memory utilization, accuracy, and power contribution, and minimum HBM needed. This database will help in various intelligent decisions such as using the most ideal hardware for an expert and dropping an expert if it does not meet the minimum required accuracy/SLO for an application.

Task-3.2: Chiplet-Based Modular Hardware Platform
After identifying the device affinity and the execution attributes of each expert, towards the goal of designing a suitable hardware platform, we envision mapping these experts onto appropriate chiplets which may differ across experts. To explore the search space, we plan to investigate four design choices: what types of chiplets are needed in a chip, how the EoE can be mapped to our chip, how the memory hierarchy can be tailored for a given EoE mapping, and how the chiplets and chips can be interconnected, as illustrated in Figure 6. We can create a chiplet version of each hardware of interest, and use them as plug-and-play modules, while combining them to create a variety of chips. Although datacenters also try to take leverage of the diversity of heterogeneous computing platforms, they are much slower and very expensive during adoption [50, 104, 105]. Instead, our chip creation strategy can quickly and frugally take advantage of upcoming accelerators not only from industrial sources but also from academic institutions. Using the expert execution attribute database, we can extract the expert-hardware affinity, corresponding to which the equivalent chiplet versions are integrated to design a chip (in this regard, we will also consider existing and upcoming hardware accelerators [47, 61, 115, 131, 173]). A chip can be *homogeneous* (containing the same chiplets like only GPUs or CPUs) or *heterogeneous* (having different types of compute chiplets including hardware support for quantization and sparsity [109]), to serve varying use-cases of EoEs. For example, an EoE consisting of GPU-affined experts can be mapped to a GPU-only chip, whereas an EoE consisting of varying chiplet affinities can be mapped to a suitable heterogeneous chip. For executing an EoE, we can explore various expert-to-chiplet mapping strategies such as one-to-one, one-to-many, many-to-one, and many-to-many, including EoE mapping to multiple chips. For example, smaller experts may be mapped to individual chiplets (or can be colocated in the same chiplet), while a large expert may be mapped to multiple chiplets. Or, multiple low hot experts can be assigned to one chiplet, and high hot experts can be distributed across chiplets. Further, neighboring experts can be mapped to adjacent chiplets for improving physical proximity and reducing data movement costs. Leveraging our prior work on multicores [23,24,77,91,112,144,183–185], we will explore compiler support in identifying suitable expert-to-chiplet mappings. Since model execution typically demands either a large memory footprint or high bandwidth or both, using both large capacity DDR and fast memory HBM memory blocks in our chip can be explored to effectively meet these memory requirements. In addition, in the cases of heavy data movement within a chiplet, the register file, scratchpad memory, and cache hierarchy can be configured for more effectiveness. Or, having a fine-grained programmer control via pragmas/annotations can help to efficiently utilize the caches. For example, in cases of large working footprint or streaming accesses, the caches can be instructed to bypass low reuse data and store only the high reuse data. To support faster inter-chip communication, which is essential for training, the chips can be interconnected using a fast fabric like NVLink [119], Infinity Fabric [13], or any newer technologies, and to cater inter-chiplet communication, which is essential for both training and inference, the chips can benefit from high-performance on-chip interconnect designs [10, 18, 31, 41, 108, 110, 116, 143]. The PI has extensive experience in designing on-chip interconnects for multicore architectures and will use the in-house simulation tools and other open-source network design tools [8, 19, 21, 116] to investigate the on-chip and inter-chip communication fabrics.

Task-3.3: Handling Unforeseen Cases using Reconfiguration
Besides having dedicated hardware for experts, there are three possible challenges which require adaption in hardware: the chiplets may incur failures, the work contained in inference is input dependent, and re-

training is needed as part of continuous learning. We plan to have "reconfigurable chiplets", which can be customized during runtime to become a specific compute engine or memory block. For example, during inference, the system can decide to map an expert onto a chip but the suitable hardware may not be directly present. In this case, the reconfigurable chiplet can be transformed into a suitable compute engine. Or, in training, new memory blocks can be spawned to support the higher KV-caching needs. During the occurrence of a fault, prior known works can be used to checkpoint and migrate the execution to a newly spawned chiplet. Further, based on our profiling of the communication behavior across chiplets and chips, we will consider performant and adaptive interconnect designs [80, 81, 110, 116, 127] to co-optimize for latency, bandwidth, and fault tolerance.

Task-3.4: Going beyond with Hardware Software Co-optimization
Achieving 100% effectiveness in SLO metrics will not be possible just with the algorithms and system techniques mentioned in Thrusts 1 and 2 unless the underlying hardware is *co-designed* with the software. In this design space, we plan to expose various "knobs" from hardware via which software can better monitor and control the execution. Recent works have started exploiting HW-SW co-optimizations, for example, application-specific software prefetching can be used to tolerate the memory access latencies [70, 71], and L2-cache can be precisely populated with most frequency accessed embeddings [70]. Further, in some specific cases of unknown constraints, the software can indicate an expert to *migrate* to another chiplet. Additionally, the hardware can expose various performance counters to the software. For example, a performance counter exposing data movement traffic in NoC can be used by runtime support to shuffle the expert to chiplet mappings. Thus, by investigating all the HW-SW co-optimizer pairs, we envision boosting the overall performance. In addition, the plug-and-play nature of chiplet provides better "fault isolation", as only the faulty chiplet can be changed as opposed to throwing away an entire chip. Since fault-tolerance has become a major concern for long-running training jobs [39, 75, 200], our proposed chiplet-based design should be able to handle hardware and software faults in a gracefully-degraded manner. In this project, we will investigate the fault-tolerance behavior of the chiplet architecture by injecting different types of faults.

2.4 Thrust-4: Evaluation And Fine Tuning

In order to evaluate the efficacy our EoE-based cross-layer design framework in terms of performance, energy efficiency and accuracy metrics, we propose to develop a comprehensive evaluation platform that consists of an in-depth simulation infrastructure, analytical models and appropriate measurements on available systems. Specifically, this thrust targets at answering the following questions: i) *What kind of simulation-emulation system is needed to carry out our experiments?*; ii) *What are the individual impacts of the optimizations discussed in Thrusts 1-3, and what is their combined effect?*; iii) *What types of accelerator-based chiplets and chips perform better for training and inference of EoEs?*; iv) *Should we build separate training and inference chiplets/chips, or would a unified chip/chiplet suffice?*; v) *What is the estimated training time given the size of input data and choice of model architecture?*; vi) *What are the desired architectural details of accelerators employed to do training/inference?*; and vii) *What are the performance, power and accuracy tradeoffs in training or inference, given hyper-parameters like the size of training dataset and the length of prompt?*

Task-4.1: Evaluation Infrastructure: Experiments + Simulation + Analytical Modeling
To compare and contrast the training complexity, training duration, accuracy, and suitability to the custom hardware (utilization) of our proposed EoE-based LLM approach, we need an integrated framework that takes into account the size of the training dataset, hyper-parameters of the smaller expert model and the given hardware resource configuration to estimate the aforementioned parameters. Given extremely long training latencies, we cannot rely on simulation alone as it would take extremely long running times. Observing this, we propose to develop an evaluation framework with three components. The first component of this framework will employ actual machine experiments on Argonne National Lab (ANL) machines (see the collaboration letter from ANL and our preliminary results [26]). These experiments will involve not only state-of-the-art GPUs but also hardware accelerators such as Groq [5], Cerebras [96], SambaNova [131], Habana Gaudi [87], and GraphCore [54] (all available on ANL machines). The second component will be based on simulation. To achieve a simulation that aligns closely with actual device implementation and captures detailed system-level interactions, we intend to employ several specialized tools (see Figure 7). Since modularity is the backbone of our proposed framework, we envision a detailed simulation of the individual hardware modules (e.g., chiplets, cache/memory components, on-chip and chip-to-chip networks,

etc), as shown in Figure 7 and stitching the evaluation framework for all such discrete modules together to build an "end-to-end" modeling platform for the proposed system. The interconnects within the System on Chip (SoC) connecting different chiplets as well as within a chiplet will be modeled using gem5's [20] on-chip network implementation, GARNET [158]. Tensor cores and matrix multiplication engines will be simulated using ScaleSim v2 [137,138], while the memory hierarchy (both DRAMs and HBMs) will be represented using Ramulator [36]. The custom cache hierarchy will be analyzed through gem5 and Cacti [114], and storage drives will be modeled using MQSim [154] and FlashSim [82]. Additionally, our custom chiplet hardware will be simulated using the publicly-available RapidChiplet [68] simulator. Since the simulation of generative AI (especially training monolithic LLMs) can take extremely long running times and may not even be feasible in some cases, we will also build, as the third component of our evaluation framework, using data from our actual machine experiments and simulations, "analytical models" that provide fast evaluation of EoE systems and LLMs with reasonable accuracy, such as [40,169]. Our approach will embody the best of both the worlds – taking the deeper system level insights from real hardware/simulators and extending it to large scale by analytically modeling their behavior as a complex system.

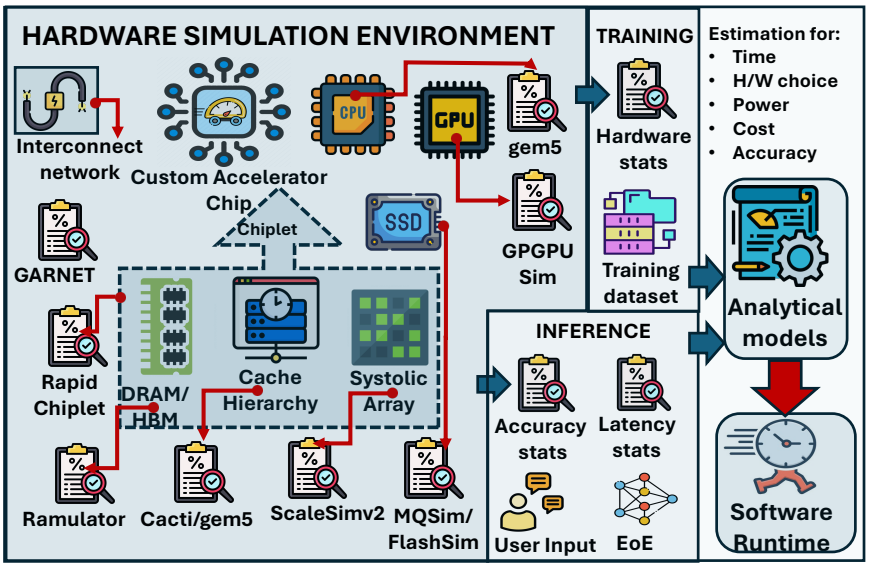

Figure 7*: An end-to-end evaluation environment*

Task-4.2: Methodology
We plan to compare our proposed optimizations against state-of-the-art MoEs [38, 42, 57, 74, 83, 89], CoEs [60, 67, 131, 151] and monolithic LLMs [107, 192] on state-of-the-art GPUs, CPUs, and custom accelerators (e.g., Groq, Cerebras, SambaNova, Habana Gaudi, and GraphCore). In our evaluation, we will employ both architecture-level and LLM/application-level metrics. The former includes execution cycles, energy consumption and carbon footprint (by extending GA4HPC [11]). In addition to these, we will also use LLM/application-level metrics such as LLMOps (an extension of MLOps [84] tailored for LLMs). To evaluate the effectiveness of our optimizations and compare them against state-of-the-art, in addition to the standalone LLM/EoE systems, we will use various applications that employ LLMs (e.g., text generation, summarization, chatbot, language translation, and sentiment analysis) as well as emerging benchmarks like HellaSwag [195], TruthfulQA [97], GLUE [159], and MMLU [191]. To gather input data, we plan to utilize publicly available datasets such as Wikipedia [164], Common Crawl [29], BookCorpus [199] and OpenWebText [49], along with our proprietary repositories. After necessary post-processing, these data points will be employed to train, validate, and evaluate our expert models.

The insights gained from these simulations and modeling efforts will be instrumental in refining runtime performance and providing informed estimates regarding the system and hardware demands of emerging applications. Furthermore, we plan to make this comprehensive full-stack development environment open-sourced, enabling the community to leverage it for informed design decision-making.

3 Related Work

LLMs have gained significant momentum in recent years and are being used in domains like virtual assistants [12,16,56], website chatbots [121], tools [48,123], notetaking/summarization [52], etc. The volume of research in LLMs is a testimonial to the interest in this area. Here, we summarize the research directly related to our proposal under the following areas:

Algorithms and Models: LLMs are known for their vast parameter sizes and training datasets, with a com-

mon belief that larger LLMs yield better performance [64]. This claim is supported by research showing the positive impact of increasing model and dataset size on accuracy [66,92]. Prior works have contributed towards reducing the parameter count and the compute complexity through model pruning [63, 98, 198], knowledge distillation [55,93,113], quantization [22,46,95] along with mixture of experts (MoE) [69], including dense MoE [37,117,126,166], sparse MoE [42,74,89,145], soft MoE [111,132,178,194], and composition of experts (CoE) [59,131,196,197]. Differentiating from prior art, our approach introduces a novel modular, collaborative framework of LLM experts, where individual experts can be trained independently and connected in various configurations. This enables flexible, diverse ways of expert adaptation and localized training to keep pace with rapidly changing knowledge and user needs, while also facilitating cross-layer optimization for system and architecture design.

Systems for LLMs: Since training giant monolithic models entail huge compute infrastructure, prior works have explored developing parallelization techniques like model parallelism [90, 149], data parallelism [118,128], and hybrid parallelism [135]. Distributed training frameworks like Horovod [141], MegatronLM [149], and DeepSpeed [65] ensure optimized data communication for distributed memory usage along with dynamic batching so that scaling overheads are minimized and resources are maximally utilized. Existing literature also examines the impact of software optimizations like kernel fusion [170] and compiler optimizations [25], performance improvements via load balancing [58, 88, 182] and resource allocation [79, 161]. In the direction of fault tolerance, there are works on various checkpointing [102, 162] strategies so that system gracefully comes out of failure with minimum loss of progress. Our approach considers the mentioned state-of-the-art optimizations and on top of that exploits the natural affinities among experts, routers, composition functions, data, and hardware to synergistically boost training/inference.

LLM Hardware and Accelerators: These sophisticated system level strategies require the use of equally high-performing hardware to complement for faster, efficient and economic implementation for both training and inferring from these models. Innovations in this domain include the latest GPUs optimized for LLM use-cases [1, 2], TPUs [76], domain-specific accelerators like Cerebras Systems Wafer-Scale Engine (WSE) [86], Graphcore Intelligence Processing Units (IPUs) [54], Habana Lab's Gaudi and Goya accelerators [87, 103], SambaNova's SN40L reconfigurable dataflow unit (RDU) [131], and Groq [61]. These aforementioned hardware equivocally echo the need of highly parallel computation with bigger and faster memory hierarchy to contain the pool of data for high-scale deployments. Our proposal to this end is to augment the said conventional wisdom with custom "chiplet-based" accelerators *tailored* for the models running on them and using their reconfigurable property to leverage the same hardware for different types of experts that we define or may emerge in the future.

Tools, Platforms and Frameworks for LLM Evaluation: Although they do not capture system insights, a few mathematical models [133] have been proposed to estimate the complexity of LLM training process. Additionally, a few simulators [9, 180] have been proposed to estimate time and cost of inferences. In this direction, we plan to use/augment these models by investigating microarchitectural, architectural and system level impacts on training and inference processes.

4 Broader Impacts

Research Ramifications: While LLMs have recently gained significant attention specifically in paving the way for Generative AI, the monolithic design of such models have made training and inference prohibitively expensive. In this context, our project takes an ambitious step to *democratize* LLM models by exploring the design space of morphable EoEs. It will lead to a more systematic, scalable, robust, customized and cost-effective LLM models for various application domains. The proposed scalable cross-layer framework will enable the exploration of novel architectural and system-level solutions in addressing the LLM design challenges. In particular, we expect each individual thrust of our project to form a "baseline" upon which further extensions and enhancements can be built. Specifically, our algorithmic layer will advance state-of-the-art in LLM models and algorithms and generate an extensible framework in which more futuristic EoE networks can be explored. Our system support will provide a framework using which researchers can conduct scalable training and inference experiments. Our architectural support will result in a search space exploration methodology that can be used to map experts to chiplets for improving execution efficiency. Finally, our evaluation infrastructure can be used for fostering new research directions not only in LLMs, but also broadly in any "transformer-based" applications like recommendation systems and multi-modal Generative AI applications, for efficient training and inferences.

Curriculum Development Activities: As we have done in our prior NSF projects, we will integrate our research results from this project with educational activities and graduate and undergraduate student training for nurturing the future workforce in science and engineering. Our curriculum development activities include the development of two courses related two this project – (i) an undergraduate course on "Generative AI" (drawing mainly from Thrust 1 of this project and focusing on training students to develop skills like language model creation other generative AI applications) that will be used for our CS curriculum and the new AI degree, and (ii) a graduate course on "System and Architectural Support for LLMs", which will draw from the contents of Thrusts 2 and 3. Also, where possible, the research material from this project will be integrated into the ML, architecture, and systems courses the PIs are regularly teaching at Penn State.

Undergraduate Involvement: We will engage undergraduates, especially those from the Penn State's Schreyer Honors College, in the planned research activities. Considering that a lot of motivated undergraduate students at Penn State are interested in ML and AI, we plan to assign well-defined projects from this research as "honors theses". We will team an undergraduate with one graduate student for regular mentoring. We have advised several Penn State honors students (including women and minorities) through our prior NSF projects. We will also target students from the Integrated Undergraduate/Graduate (IUG) program at Penn State (which allows students to earn both a bachelor's and master's degree in five years), to get involved in Generative AI research for possible graduate studies.

Industry Collaboration: We have several ongoing collaborations with industries like NVIDIA, AMD, Google and Meta, who are major players in advancing deep learning technology/systems. We plan to collaborate with them on various aspects of this project as well, and explore opportunities for technology transfer. Furthermore, the PIs will leverage additional connections through their former students working in these companies. The feedback from industry will help us fine tune our proposed EoE models.

K12 and Outreach: Our outreach plans include involvement of underrepresented groups in computer science and engineering and various K-12 related activities. A detailed description of our **BPC plan** is included as a supplementary document. One example is the Science-U camp at Penn State, which is designed to take K-12 students through a one-week journey that investigates an area of STEM in an exciting way. We will also outreach to researchers in other disciplines by giving project-related talks at different departments at Penn State (e.g., math and statistics). The PIs are involved in several K-12 activities such as the summer CS program for girls (funded by CSE and led by Das). We plan to continue the summer program involving more schools and students in coming years, where we will expose them to Generative AI and related concepts. We will also collaborate with the Penn State College of Education's CSATS (Center for Science and the Schools) to participate in the university's continuing outreach initiatives focused on STEM subjects. The PIs have supervised several women PhD students, and are currently advising a total of 6 female PhD students in their groups. The PIs plan to recruit new women and minorities for this project as well.

5 Results from Prior NSF Support

Award # 1763681 (SHF: Medium: Embracing Architectural Heterogeneity through Hardware-Software Co-design); PI: Das; Co-PIs: Sivasubramaniam and Kandemir; duration= 06/01/2018–05/31/2023; amount= $1,000,000. Intellectual Merit: This project explores hw-sw support to transform applications into suitable device-agnostic codelets, that serve as the granularity for seamless scheduling and execution across GPUs and FPGAs. Broader Impacts: The research results from this project have been fused into different classes at Penn State, and the project has benefited from the participation of undergraduate honor students. Five PhD students, supported through this project, have graduated and one has joined as a faculty. Major Results: The main research results so far from this project appear in [70, 71, 129, 140, 150, 153, 156, 174, 175, 179, 190].

Award # 2338418 (CAREER: Trustworthy Human-Centered Summarization); PI: Zhang; duration=09/15/24-08/31/29; amount=$546,000. Intellectual Merit: This research advances trustworthy summarization by centering design, development, and deployment on humans in terms of user preferences for controllability, social perspectives for fairness, and human knowledge for factuality. Broader Impacts: This project initiates several aspiring education and outreach activities supported by project research outcomes to involve, mentor, and empower female, underrepresented, disabled, and interdisciplinary students. Major Results: No publication yet as the project started in September 2024.

References

[1] NVIDIA A100 Tensor Core GPU., 2023. https://www.nvidia.com/en-us/data-center/a100/.

[2] NVIDIA H100 Tensor Core GPU, 2023. https://www.nvidia.com/en-us/data-center/technologies/hopper-architecture/.

[3] Exploring llms - real-world case studies in ai-generated art & literature. https://www.tome01.com/exploring-llms-real-world-case-studies-in-ai-generated-art-literature, 2024. Accessed: 2024-10-22.

[4] The state of ai in early 2024: Gen ai adoption spikes and starts to generate value. https://www.mckinsey.com/capabilities/quantumblack/our-insights/the-state-of-ai, February 2024. Accessed: 2023-10-20.

[5] Dennis Abts, Garrin Kimmell, Andrew Ling, John Kim, Matt Boyd, Andrew Bitar, Sahil Parmar, Ibrahim Ahmed, Roberto DiCecco, David Han, John Thompson, Michael Bye, Jennifer Hwang, Jeremy Fowers, Peter Lillian, Ashwin Murthy, Elyas Mehtabuddin, Chetan Tekur, Thomas Sohmers, Kris Kang, Stephen Maresh, and Jonathan Ross. A software-defined tensor streaming multiprocessor for large-scale machine learning. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*, pages 567–580, 2022.

[6] Eleni Adamopoulou and Lefteris Moussiades. An overview of chatbot technology. In *IFIP international conference on artificial intelligence applications and innovations*, pages 373–383. Springer, 2020.

[7] Muhammad Adnan, Akhil Arunkumar, Gaurav Jain, Prashant Nair, Ilya Soloveychik, and Purushotham Kamath. Keyformer: Kv cache reduction through key tokens selection for efficient generative inference. In P. Gibbons, G. Pekhimenko, and C. De Sa, editors, *Proceedings of Machine Learning and Systems*, volume 6, pages 114–127, 2024.

[8] Niket Agarwal, Tushar Krishna, Li-Shiuan Peh, and Niraj K Jha. Garnet: A detailed on-chip network model inside a full-system simulator. In *2009 IEEE international symposium on performance analysis of systems and software*, pages 33–42. IEEE, 2009.

[9] Amey Agrawal, Nitin Kedia, Jayashree Mohan, Ashish Panwar, Nipun Kwatra, Bhargav Gulavani, Ramachandran Ramjee, and Alexey Tumanov. Vidur: A large-scale simulation framework for llm inference. *Proceedings of Machine Learning and Systems*, 6:351–366, 2024.

[10] Fawaz Alazemi, Arash Azizimazreah, Bella Bose, and Lizhong Chen. Routerless network-on-chip. In *2018 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pages 492–503. IEEE, 2018.

[11] Green Algorithms. Green Algorithms 4 HPC, August 2024.

[12] Amazon. Alexa. "https://developer.amazon.com/en-US/alexa/alexa-ai", 2024.

[13] AMD. Infinity fabric. "https://www.amd.com/content/dam/amd/en/documents/instinct-tech-docs/data-sheets/amd-instinct-mi300x-platform-data-sheet.pdf", 2024.

[14] Lasse F Wolff Anthony, Benjamin Kanding, and Raghavendra Selvan. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. *arXiv preprint arXiv:2007.03051*, 2020.

[15] Apache Software Foundation. Apache® Subversion®. https://subversion.apache.org/, 2024.

[16] Apple. Apple intelligence. "https://machinelearning.apple.com/research/introducing-apple-foundation-models", 2024.

[17] Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623, 2021.

[18] Maciej Besta, Syed Minhaj Hassan, Sudhakar Yalamanchili, Rachata Ausavarungnirun, Onur Mutlu, and Torsten Hoefler. Slim noc: A low-diameter on-chip network topology for high energy efficiency and scalability. *ACM SIGPLAN Notices*, 53(2):43–55, 2018.

[19] Srikant Bharadwaj, Jieming Yin, Bradford Beckmann, and Tushar Krishna. Kite: A family of heterogeneous interposer topologies enabled via accurate interconnect modeling. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[20] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, August 2011.

[21] Jingwei Cai, Zuotong Wu, Sen Peng, Yuchen Wei, Zhanhong Tan, Guiming Shi, Mingyu Gao, and Kaisheng Ma. Gemini: Mapping and architecture co-exploration for large-scale dnn chiplet accelerators. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 156–171. IEEE, 2024.

[22] Jerry Chee, Yaohui Cai, Volodymyr Kuleshov, and Christopher M De Sa. Quip: 2-bit quantization of large language models with guarantees. *Advances in Neural Information Processing Systems*, 36, 2024.

[23] Guangyu Chen and Feihui Li. Application mapping for chip multiprocessors. In *Proceedings of the 45th annual design automation conference*, pages 620–625, 2008.

[24] Guangyu Chen, Feihui Li, and Mahmut Kandemir. Compiler-directed application mapping for noc based chip multiprocessors. *ACM SIGPLAN Notices*, 42(7):155–157, 2007.

[25] Tianqi Chen, Benjamin Moreau, Chunting Zheng, Yutian Tang, Zijian Yan, Yanan Song, Yuhao Jia, Maximilian Seeger, Lingfeng Wang, and Hai Bian. Tvm: An automated end-to-end optimizing compiler for deep learning. *Proceedings of the ACM on Programming Languages*, 2(4):1–25, 2018.

[26] Scott Cheng, Jun-Liang Lin, Murali Emani, Siddhisanket Raskar, Sam Foreman, Zhen Xie, Venkatram Vishwanath, and Mahmut Taylan Kandemir. Thorough characterization and analysis of large transformer model training at-scale. *Proc. ACM Meas. Anal. Comput. Syst.*, 8(1), February 2024.

[27] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[28] Roi Cohen, Eden Biran, Ori Yoran, Amir Globerson, and Mor Geva. Evaluating the ripple effects of knowledge editing in language models. *Transactions of the Association for Computational Linguistics*, 12:283–298, 2024.

[29] Common Crawl. Common crawl corpus. `https://commoncrawl.org`, 2024. Accessed: 2024-10-18.

[30] Bill Dally. Directions in deep learning hardware. YouTube, 2024. Available at: https://youtu.be/gofI47kfD28?t=685 [Accessed: 10/24/2024].

[31] Reetuparna Das, Onur Mutlu, Thomas Moscibroda, and Chita R Das. Aergia: Exploiting packet latency slack in on-chip networks. *ACM SIGARCH computer architecture news*, 38(3):106–116, 2010.

[32] Sarkar Snigdha Sarathi Das, Arzoo Katiyar, Rebecca Passonneau, and Rui Zhang. CONTaiNER: Few-shot named entity recognition via contrastive learning. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6338–6353, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[33] Sarkar Snigdha Sarathi Das, Ranran Haoran Zhang, Peng Shi, Wenpeng Yin, and Rui Zhang. Unified low-resource sequence labeling by sample-aware dynamic sparse finetuning. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6998–7010, Singapore, December 2023. Association for Computational Linguistics.

[34] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms. *Advances in Neural Information Processing Systems*, 36, 2024.

[35] Shichen Dong, Wen Cheng, Jiayu Qin, and Wei Wang. Qaq: Quality adaptive quantization for llm kv cache, 2024.

[36] Xiaobin Dong, Kurt Keutzer, and Cong Zhang. Ramulator: A cycle accurate dram simulator. In *Proceedings of the 22nd ACM International Symposium on High-Performance Parallel and Distributed Computing*, pages 151–160. ACM, 2014.

[37] Shihan Dou, Enyu Zhou, Yan Liu, Songyang Gao, Jun Zhao, Wei Shen, Yuhao Zhou, Zhiheng Xi, Xiao Wang, Xiaoran Fan, Shiliang Pu, Jiang Zhu, Rui Zheng, Tao Gui, Qi Zhang, and Xuanjing Huang. The art of balancing: Revolutionizing mixture of experts for maintaining world knowledge in language model alignment. 2023.

[38] Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Maxim Krikun, Yanqi Zhou, Adams Wei Yu, Orhan Firat, Barret Zoph, Liam Fedus, Maarten Bosma, Zongwei Zhou, Tao Wang, Yu Emma Wang, Kellie Webster, Marie Pellat, Kevin Robinson, Kathy Meier-Hellstern, Toju Duke, Lucas Dixon, Kun Zhang, Quoc V. Le, Yonghui Wu, Zhifeng Chen, and Claire Cui. Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, pages 5547–5569. PMLR, 2022.

[39] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Lauren Rantala-Yeary,

Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaoqing Ellen Tan, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aaron Grattafiori, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alex Vaughan, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Franco, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, Danny Wyatt, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Firat Ozgenel, Francesco Caggioni, Francisco Guzmán, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Govind Thattai, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Karthik Prasad, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kun Huang, Kunal Chawla, Kushal Lakhotia, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Maria Tsimpoukelli, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikolay Pavlovich Laptev, Ning Dong, Ning Zhang, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Raymond Li, Rebekkah

Hogan, Robin Battey, Rocky Wang, Rohan Maheswari, Russ Howes, Ruty Rinott, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Kohler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vítor Albiero, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaofang Wang, Xiaojian Wu, Xiaolan Wang, Xide Xia, Xilun Wu, Xinbo Gao, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yuchen Hao, Yundi Qian, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, and Zhiwei Zhao. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[40] Murali Emani, Sam Foreman, Varuni Sastry, Zhen Xie, Siddhisanket Raskar, William Arnold, Rajeev Thakur, Venkatram Vishwanath, Michael E. Papka, Sanjif Shanmugavelu, Darshan Gandhi, Hengyu Zhao, Dun Ma, Kiran Ranganath, Rick Weisner, Jiunn-yeu Chen, Yuting Yang, Natalia Vassilieva, Bin C. Zhang, Sylvia Howland, and Alexander Tsyplikhin. Toward a holistic performance evaluation of large language models across diverse ai accelerators. In *2024 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1–10, 2024.

[41] Chris Fallin, Chris Craik, and Onur Mutlu. Chipper: A low-complexity bufferless deflection router. In *2011 IEEE 17th International Symposium on High Performance Computer Architecture*, pages 144–155. IEEE, 2011.

[42] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[43] Yuan Feng, Junlin Lv, Yukun Cao, Xike Xie, and S. Kevin Zhou. Ada-kv: Optimizing kv cache eviction by adaptive budget allocation for efficient llm inference, 2024.

[44] Financial Times. Amazon buys stake in nuclear energy developer in push to power data centres. `https://www.ft.com/content/00776191-b010-4104-add4-8dc430386911`, 2024. Accessed: 2024-10-20.

[45] Giorgio Franceschelli and Mirco Musolesi. On the creativity of large language models. *AI Models FYI*, 2024. Available at: `https://www.aimodels.fyi`.

[46] Elias Frantar and Dan Alistarh. Qmoe: Practical sub-1-bit compression of trillion-parameter models. *arXiv preprint arXiv:2310.16795*, 2023.

[47] Gerasimos Gerogiannis, Sriram Aananthakrishnan, Josep Torrellas, and Ibrahim Hur. Hottiles: Accelerating spmm with heterogeneous accelerator architectures. In *2024 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, pages 1012–1028. IEEE, 2024.

[48] Github. Github copilot. "`https://github.com/features/copilot`", 2024.

[49] Mohammad Gokaslan, Girish Mishra, and Andrea Madotto. Openwebtext corpus: Extracting webcontent using reddit links. *arXiv preprint arXiv:2001.08023*, 2020.

[50] Google. Announcing trillium, the sixth generation of google cloud tpu, 2024.

[51] Google. Google gemini. "`https://gemini.google.com/app`", 2024.

[52] Google. Google notebooklm. "`https://notebooklm.google/`", 2024.

[53] Google Blog. Google and kairos power nuclear energy agreement. https://blog.google/outreach-initiatives/sustainability/google-kairos-power-nuclear-energy-agreement/, 2024. Accessed: 2024-10-20.

[54] Graphcore. Graphcore intelligence processing unit (ipu). https://www.graphcore.ai/products/ipu, 2023. Accessed: 2024-04-27.

[55] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024.

[56] Yanchu Guan, Dong Wang, Zhixuan Chu, Shiyu Wang, Feiyue Ni, Ruihua Song, Longfei Li, Jinjie Gu, and Chenyi Zhuang. Intelligent virtual assistants with llm-based process automation, 2023.

[57] Nikhil Gupta and Jason Yip. Dbrx: Creating an llm from scratch using databricks. In *Databricks Data Intelligence Platform: Unlocking the GenAI Revolution*, pages 311–330. Springer, 2024.

[58] Rakesh Gupta, Anil Singh, and Deepak Kumar. Adaptive load balancing in distributed machine learning systems. *Journal of Parallel and Distributed Computing*, 145:45–58, 2020.

[59] Suchin Gururangan, Mike Lewis, Ari Holtzman, Noah A Smith, and Luke Zettlemoyer. Demix layers: Disentangling domains for modular language modeling. *arXiv preprint arXiv:2108.05036*, 2021.

[60] Suchin Gururangan, Margaret Li, Mike Lewis, Weijia Shi, Tim Althoff, Noah A Smith, and Luke Zettlemoyer. Scaling expert language models with unsupervised domain discovery. *arXiv preprint arXiv:2303.14177*, 2023.

[61] Linley Gwennap. Groq rocks neural networks. *Microprocessor Report, Tech. Rep., jan*, 2020.

[62] Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alex Wardle-Solano, Hannah Szabo, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander R. Fabbri, Wojciech Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. Folio: Natural language reasoning with first-order logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024.

[63] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *Journal of Machine Learning Research*, 22(241):1–124, 2021.

[64] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[65] Connor Holmes, Masahiro Tanaka, Michael Wyatt, Ammar Ahmad Awan, Jeff Rasley, Samyam Rajbhandari, Reza Yazdani Aminabadi, Heyang Qin, Arash Bakhtiari, Lev Kurilenko, and Yuxiong He. Deepspeed-fastgen: High-throughput text generation for llms via mii and deepspeed-inference. *arXiv preprint arXiv:2401.08671*, 2024.

[66] Lei Huang, Weijiang Yu, Weitao Ma, Weihong Zhong, Zhangyin Feng, Haotian Wang, Qianglong Chen, Weihua Peng, Xiaocheng Feng, Bing Qin, and Ting Liu. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *arXiv preprint arXiv:2311.05232*, 2023.

[67] Shaomang Huang, Jianfeng Pan, and Hanzhong Zheng. Ccoe: A compact llm with collaboration of experts. *arXiv preprint arXiv:2407.11686*, 2024.

[68] Patrick Iff, Benigna Bruggmann, Maciej Besta, Luca Benini, and Torsten Hoefler. Rapidchiplet: A toolchain for rapid design space exploration of chiplet architectures, 2023.

[69] Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

[70] Rishabh Jain, Vivek M Bhasi, Adwait Jog, Anand Sivasubramaniam, Mahmut Taylan Kandemir, and Chita R Das. Pushing the performance envelope of dnn-based recommendation systems inference on gpus. In *To be presented in proceedings of the 57th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 62–76, 2024.

[71] Rishabh Jain, Scott Cheng, Vishwas Kalagi, Vrushabh Sanghavi, Samvit Kaul, Meena Arunachalam, Kiwan Maeng, Adwait Jog, Anand Sivasubramaniam, Mahmut Taylan Kandemir, and Chita R. Das. Optimizing cpu performance for recommendation systems at-scale. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–15, 2023.

[72] Natalie Enright Jerger, Ajaykumar Kannan, Zimo Li, and Gabriel H Loh. Noc architectures for silicon interposer systems: Why pay for more wires when you can get them (from your interposer) for free? In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 458–470. IEEE, 2014.

[73] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[74] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[75] Ziheng Jiang, Haibin Lin, Yinmin Zhong, Qi Huang, Yangrui Chen, Zhi Zhang, Yanghua Peng, Xiang Li, Cong Xie, Shibiao Nong, Yulu Jia, Sun He, Hongmin Chen, Zhihao Bai, Qi Hou, Shipeng Yan, Ding Zhou, Yiyao Sheng, Zhuo Jiang, Haohan Xu, Haoran Wei, Zhang Zhang, Pengfei Nie, Leqi Zou, Sida Zhao, Liang Xiang, Zherui Liu, Zhe Li, Xiaoying Jia, Jianxi Ye, Xin Jin, and Xin Liu. {MegaScale}: Scaling large language model training to more than 10,000 {GPUs}. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 745–760, 2024.

[76] Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Clifford Young, Xiang Zhou, Zongwei Zhou, and David A Patterson. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, New York, NY, USA, 2023. Association for Computing Machinery.

[77] Mahmut Kandemir, Taylan Yemliha, SaiPrashanth Muralidhara, Shekhar Srikantaiah, Mary Jane Irwin, and Yuanrui Zhnag. Cache topology aware computation mapping for multicores. In *Proceedings of the 31st ACM SIGPLAN Conference on Programming Language Design and Implementation*, pages 74–85, 2010.

[78] Keboola. Keboola, the platform that automates data lineage. `https://www.keboola.com/product/data-lineage`. (Accessed on 09/12/2023).

[79] Jihye Kim, Sungho Lee, and Minseok Park. Optimizing resource allocation in gpu clusters for deep learning training. *IEEE Transactions on Parallel and Distributed Systems*, 32(8):1987–2000, 2021.

[80] Jongman Kim, Chrysostomos Nicopoulos, Dongkook Park, Vijaykrishnan Narayanan, Mazin S Yousif, and Chita R Das. A gracefully degrading and energy-efficient modular router architecture for on-chip networks. *ACM SIGARCH Computer Architecture News*, 34(2):4–15, 2006.

[81] Jongman Kim, Dongkook Park, and Chita R Das. A low latency router supporting adaptivity for on-chip interconnects. In *Proceedings of the 42nd annual Design Automation Conference*, pages 559–564, 2005.

[82] Youngjae Kim, Brendan Tauras, Aayush Gupta, and Bhuvan Urgaonkar. Flashsim: A simulator for nand flash-based solid-state drives. In *2009 First International Conference on Advances in System Simulation*, pages 125–131, 2009.

[83] Yeskendir Koishekenov, Alexandre Berard, and Vassilina Nikoulina. Memory-efficient nllb-200: Language-specific expert pruning of a massively multilingual machine translation model. *arXiv preprint arXiv:2212.09811*, 2022.

[84] Dominik Kreuzberger, Niklas Kühl, and Sebastian Hirschl. Machine learning operations (mlops): Overview, definition, and architecture. *IEEE Access*, 11:31866–31879, 2023.

[85] Harsh Kumar, Jonathan Vincentius, Ewan Jordan, and Ashton Anderson. Human creativity in the age of llms: Randomized experiments on divergent and convergent thinking. *arXiv*, arXiv:2410.03703, 2024. Available at: `https://doi.org/10.48550/arXiv.2410.03703`.

[86] Mandy La and Andrew Chien. Cerebras systems: Journey to the wafer-scale engine. *University of Chicago, Tech. Rep*, 2020.

[87] Habana Labs. Habana gaudi ai processor. `https://habana.ai/products/gaudi2/`, 2023. Accessed: 2024-04-27.

[88] Minho Lee, Sooyeon Park, and Hyunsoo Choi. Load balancing techniques for efficient training of large-scale neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(6):2345–2356, 2021.

[89] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

[90] Dmitry Lepikhin, HyoukJoong Lee, Yuanzhong Xu, Dehao Chen, Orhan Firat, Yanping Huang, Maxim Krikun, Noam Shazeer, and Zhifeng Chen. Gshard: Scaling giant models with conditional computation and automatic sharding. *arXiv preprint arXiv:2006.16668*, 2020.

[91] Feihui Li, Chrysostomos Nicopoulos, Thomas Richardson, Yuan Xie, Vijaykrishnan Narayanan, and Mahmut Kandemir. Design and management of 3d chip multiprocessors using network-in-memory. *ACM SIGARCH Computer Architecture News*, 34(2):130–141, 2006.

[92] Junyi Li, Jie Chen, Ruiyang Ren, Xiaoxue Cheng, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. The dawn after the dark: An empirical study on factuality hallucination in large language models. *arXiv preprint arXiv:2401.03205*, 2024.

[93] Lei Li, Yankai Lin, Shuhuai Ren, Peng Li, Jie Zhou, and Xu Sun. Dynamic knowledge distillation for pre-trained language models. *arXiv preprint arXiv:2109.11295*, 2021.

[94] Pengfei Li, Jianyi Yang, Mohammad A. Islam, and Shaolei Ren. Making ai less "thirsty": Uncovering and addressing the secret water footprint of ai models. *arXiv preprint arXiv:2304.03271*, April 2023. Accessed: 2023-10-20.

[95] Shiyao Li, Xuefei Ning, Ke Hong, Tengxuan Liu, Luning Wang, Xiuhong Li, Kai Zhong, Guohao Dai, Huazhong Yang, and Yu Wang. Llm-mq: Mixed-precision quantization for efficient llm deployment. In *The Efficient Natural Language and Speech Processing Workshop with NeurIPS*, volume 9, 2023.

[96] Sean Lie. Cerebras architecture deep dive: First look inside the hardware/software co-design for deep learning. *IEEE Micro*, 43(3):18–30, 2023.

[97] Zi Lin, Diana Jin, and Saurabh Singh. Truthfulqa: Measuring how models mimic human falsehoods. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 465–482. ACM, 2022.

[98] Zhuang Liu, Mingjie Sun, Tinghui Zhou, Gao Huang, and Trevor Darrell. Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*, 2018.

[99] Zichang Liu, Aditya Desai, Fangshuo Liao, Weitao Wang, Victor Xie, Zhaozhuo Xu, Anastasios Kyrillidis, and Anshumali Shrivastava. Scissorhands: exploiting the persistence of importance hypothesis for llm kv cache compression at test time. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2024. Curran Associates Inc.

[100] Andrew J. Lohn and Micah Musser. Ai and compute: How much longer can computing power drive artificial intelligence progress ? `https://cset.georgetown.edu/wp-content/uploads/AI-and-Compute-How-Much-Longer-Can-Computing-Power-Drive-Artificial-Intelligence-Progress.pdf`, January 2022. Accessed: 2023-10-20.

[101] Sadhika Malladi, Tianyu Gao, Eshaan Nichani, Alex Damian, Jason D Lee, Danqi Chen, and Sanjeev Arora. Fine-tuning language models with just forward passes. *Advances in Neural Information Processing Systems*, 36:53038–53075, 2023.

[102] Avinash Maurya, Robert Underwood, M. Mustafa Rafique, Franck Cappello, and Bogdan Nicolae. Datastates-llm: Lazy asynchronous checkpointing for large language models. In *Proceedings of the 33rd International Symposium on High-Performance Parallel and Distributed Computing*, HPDC '24, page 227–239, New York, NY, USA, 2024. Association for Computing Machinery.

[103] Eitan Medina and Eran Dagan. Habana labs purpose-built ai inference and training processor architectures: Scaling ai training systems using standard ethernet with gaudi processor. *IEEE Micro*, 40(2):17–24, 2020.

[104] Meta. Building meta's genai infrastructure, 2024.

[105] Meta. Our next-generation meta training and inference accelerator, 2024.

[106] Meta AI. LLaMA-3.1 405B: Pre-trained Language Model. Hugging Face Model Repository, 2023. Available at: `https://huggingface.co/meta-llama/Llama-3.1-405B`.

[107] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao. Large language models: A survey. *arXiv preprint arXiv:2402.06196*, 2024.

[108] Asit K Mishra, Xiangyu Dong, Guangyu Sun, Yuan Xie, Narayanan Vijaykrishnan, and Chita R Das. Architecting on-chip interconnects for stacked 3d stt-ram caches in cmps. *ACM SIGARCH Computer Architecture News*, 39(3):69–80, 2011.

[109] Asit K. Mishra, Jorge Albericio Latorre, Jeff Pool, Darko Stosic, Dusan Stosic, Ganesh Venkatesh, Chong Yu, and Paulius Micikevicius. Accelerating sparse deep neural networks. *CoRR*, abs/2104.08378, 2021.

[110] Asit K Mishra, Narayanan Vijaykrishnan, and Chita R Das. A case for heterogeneous on-chip interconnects for cmps. *ACM SIGARCH Computer Architecture News*, 39(3):389–400, 2011.

[111] Mohammed Muqeeth, Haokun Liu, and Colin Raffel. Soft merging of experts with adaptive routing. *arXiv preprint arXiv:2306.03745*, 2023.

[112] Sai Prashanth Muralidhara, Lavanya Subramanian, Onur Mutlu, Mahmut Kandemir, and Thomas Moscibroda. Reducing memory interference in multicore systems via application-aware memory channel partitioning. In *Proceedings of the 44th annual IEEE/ACM international symposium on microarchitecture*, pages 374–385, 2011.

[113] Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation. *arXiv preprint arXiv:2407.14679*, 2024.

[114] Naveen Muralimanohar, Rajeev Balasubramonian, and Norman P. Jouppi. Cacti 6.0: A tool to model large caches. 2009.

[115] Abhinand Nasari, Lujun Zhai, Zhenhua He, Hieu Le, Suxia Cui, Dhruva Chakravorty, Jian Tao, and Honggao Liu. Porting ai/ml models to intelligence processing units (ipus). In *Practice and Experience in Advanced Research Computing 2023: Computing for the Common Good*, PEARC '23, page 231–236, New York, NY, USA, 2023. Association for Computing Machinery.

[116] Chrysostomos A Nicopoulos, Dongkook Park, Jongman Kim, Mazin S Yousif, and Chita R Das. Vichar: A dynamic virtual channel regulator for network-on-chip routers. In *2006 39th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO'06)*, pages 333–346. IEEE, 2006.

[117] Xiaonan Nie, Xupeng Miao, Shijie Cao, Lingxiao Ma, Qibin Liu, Jilong Xue, Youshan Miao, Yi Liu, Zhi Yang, and Bin Cui. Evomoe: An evolutional mixture-of-experts training framework via dense-to-sparse gate. *arXiv preprint arXiv:2112.14397*, 2021.

[118] NVIDIA. Nvidia's deep learning data parallelism strategies for high-performance training, 2021.

[119] Nvidia. Nvidia nvlink. "`https://www.nvidia.com/en-us/data-center/nvlink/`", 2024.

[120] Octopai. Octopai: Automated data lineage, data catalog and discovery. `https://www.octopai.com/`. (Accessed on 09/12/2023).

[121] Julius Odede and Ingo Frommholz. Jaybot–aiding university students and admission with an llm-based chatbot. In *Proceedings of the 2024 Conference on Human Information Interaction and Retrieval*, pages 391–395, 2024.

[122] Journal of the American Medical Informatics Association. Efficient healthcare with large language models: optimizing clinical workflow and enhancing patient care. *Oxford Academic*, 2024.

[123] OpenAI. Openai chatgpt. "`https://openai.com/chatgpt/overview/`", 2024.

[124] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue,

Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[125] Will Orwig and Daniel Schacter. Creative writing in humans and large language models. In *Harvard Brain Science Initiative*, Boston, MA, 2024. Available at: `https://brain.harvard.edu`.

[126] Bowen Pan, Yikang Shen, Haokun Liu, Mayank Mishra, Gaoyuan Zhang, Aude Oliva, Colin Raffel, and Rameswar Panda. Dense training, sparse inference: Rethinking training of mixture-of-experts language models. *arXiv preprint arXiv:2404.05567*, 2024.

[127] Dongkook Park, Chrysostomos Nicopoulos, Jongman Kim, Narayanan Vijaykrishnan, and Chita R Das. Exploring fault-tolerant network-on-chip architectures. In *International Conference on Dependable Systems and Networks (DSN'06)*, pages 93–104. IEEE, 2006.

[128] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. `https://pytorch.org/`, 2019. Accessed: 2024-04-27.

[129] Ashutosh Pattnaik, Xulong Tang, Onur Kayiran, Adwait Jog, Asit Mishra, Mahmut T. Kandemir, Anand Sivasubramaniam, and Chita R. Das. Opportunistic computing in gpu architectures. In *Proceedings of the 46th International Symposium on Computer Architecture*, ISCA '19, page 210–223, New York, NY, USA, 2019. Association for Computing Machinery.

[130] Wei Peng. Large language models in healthcare and medical domain: A review. *Informatics*, 11(3):57, 2024.

[131] Raghu Prabhakar, Ram Sivaramakrishnan, Darshan Gandhi, Yun Du, Mingran Wang, Xiangyu Song, Kejie Zhang, Tianren Gao, Angela Wang, Karen Li, Yongning Sheng, Joshua Brot, Denis Sokolov, Apurv Vivek, Calvin Leung, Arjun Sabnis, Jiayu Bai, Tuowen Zhao, Mark Gottscho, David Jackson, Mark Luttrell, Manish K. Shah, Edison Chen, Kaizhao Liang, Swayambhoo Jain, Urmish Thakker, Dawei Huang, Sumti Jairath, Kevin J. Brown, and Kunle Olukotun. Sambanova sn40l: Scaling the ai memory wall with dataflow and composition of experts. *arXiv preprint arXiv:2405.07518*, 2024.

[132] Joan Puigcerver, Carlos Riquelme Ruiz, Basil Mustafa, and Neil Houlsby. From Sparse to Soft Mixtures of Experts. In *The Twelfth International Conference on Learning Representations*, 2023.

[133] Zhenting Qi, Hongyin Luo, Xuliang Huang, Zhuokai Zhao, Yibo Jiang, Xiangjun Fan, Himabindu Lakkaraju, and James Glass. Quantifying generalization complexity for large language models, 2024.

[134] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, and Yuxiong He. Deepspeed-moe: Advancing mixture-of-experts inference and training to power next-generation ai scale. In *International conference on machine learning*, pages 18332–18346. PMLR, 2022.

[135] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.

[136] Reuters. Meta strikes geothermal energy deal with sage geosystems to power data centers. `https://www.reuters.com/business/energy/meta-strikes-geothermal-energy-deal-with-sage-geosystems-power-data-centers-2024-08-26/`, 2024. Accessed: 2024-10-20.

[137] Ananda Samajdar, Jan Moritz Joseph, Yuhao Zhu, Paul Whatmough, Matthew Mattina, and Tushar Krishna. A systematic methodology for characterizing scalability of dnn accelerators using scale-sim. In *2020 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 58–68. IEEE, 2020.

[138] Ananda Samajdar, Yuhao Zhu, Paul Whatmough, Matthew Mattina, and Tushar Krishna. Scale-sim: Systolic cnn accelerator simulator. *arXiv preprint arXiv:1811.02883*, 2018.

[139] Warren S Sarle. Finding groups in data: An introduction to cluster analysis., 1991.

[140] Anup Sarma, Sonali Singh, Huaipan Jiang, Rui Zhang, Mahmut Kandemir, and Chita Das. Structured in space, randomized in time: Leveraging dropout in rnns for efficient training. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, volume 34, pages 24545–24555. Curran Associates, Inc., 2021.

[141] Alexander Sergeev and Mike Del Balso. Horovod: fast and easy distributed deep learning in tensorflow, 2018.

[142] Shaip. Large language models in healthcare: Breakthroughs, use cases, and challenges. *Shaip*, 2024.

[143] Akbar Sharifi, Emre Kultursay, Mahmut Kandemir, and Chita R Das. Addressing end-to-end memory access latency in noc-based multicores. In *2012 45th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 294–304. IEEE, 2012.

[144] Akbar Sharifi, Shekhar Srikantaiah, Asit K Mishra, Mahmut Kandemir, and Chita R Das. Mete: meeting end-to-end qos in multicores through system-wide resource management. In *Proceedings of the ACM SIGMETRICS joint international conference on Measurement and modeling of computer systems*, pages 13–24, 2011.

[145] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[146] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR, 2023.

[147] Haizhou Shi, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, and Hao Wang. Continual learning of large language models: A comprehensive survey. *arXiv preprint arXiv:2404.16789*, 2024.

[148] Luohe Shi, Hongyi Zhang, Yao Yao, Zuchao Li, and Hai Zhao. Keep the cost down: A review on methods to optimize llm' s kv-cache consumption, 2024.

[149] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-lm: Training multi-billion parameter language models using model parallelism, 2020.

[150] Sonali Singh, Anup Sarma, Sen Lu, Abhronil Sengupta, Mahmut T. Kandemir, Emre Neftci, Vijaykrishnan Narayanan, and Chita R. Das. Skipper: Enabling efficient snn training through activation-checkpointing and time-skipping. In *2022 55th IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 565–581, 2022.

[151] Sainbayar Sukhbaatar, Olga Golovneva, Vasu Sharma, Hu Xu, Xi Victoria Lin, Baptiste Rozière, Jacob Kahn, Daniel Li, Wen tau Yih, Jason Weston, and Xian Li. Branch-train-mix: Mixing expert llms into a mixture-of-experts llm. *arXiv preprint arXiv:2403.07816*, 2024.

[152] Masahiro Sunohara, Takayuki Tokunaga, Takashi Kurihara, and Mitsutoshi Higashi. Silicon interposer with tsvs (through silicon vias) and fine multilayer wiring. In *2008 58th Electronic Components and Technology Conference*, pages 847–852. IEEE, 2008.

[153] Xulong Tang, Ashutosh Pattnaik, Onur Kayiran, Adwait Jog, Mahmut Taylan Kandemir, and Chita Das. Quantifying data locality in dynamic parallelism in gpus. *Proc. ACM Meas. Anal. Comput. Syst.*, 2(3), December 2018.

[154] Arash Tavakkol, Juan Gómez-Luna, Mohammad Sadrosadati, Saugata Ghose, and Onur Mutlu. MQSim: A framework for enabling realistic studies of modern Multi-Queue SSD devices. In *16th USENIX Conference on File and Storage Technologies (FAST 18)*, pages 49–66, Oakland, CA, February 2018. USENIX Association.

[155] The New York Times. Amazon, google, microsoft turn to nuclear energy. `https://www.nytimes.com/2024/10/16/business/energy-environment/amazon-google-microsoft-nuclear-energy.html`, 2024. Accessed: 2024-10-20.

[156] Prashanth Thinakaran, Jashwant Raj Gunasekaran, Bikash Sharma, Mahmut Taylan Kandemir, and Chita R. Das. Kube-knots: Resource harvesting through dynamic container orchestration in gpu-based datacenters. In *2019 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 1–13, 2019.

[157] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[158] Shyam Venkataramani, Suresh Cherian, Sergio DeRose, Karthik Sankaralingam, and Sek Ching Wong. Garnet: A detailed network-on-chip simulator. In *Proceedings of the 2012 International Symposium on High Performance Computer Architecture (HPCA)*, pages 507–518. IEEE, 2012.

[159] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.

[160] Bo Wang, Maria Liakata, Arkaitz Zubiaga, and Rob Procter. A hierarchical topic modelling approach for tweet clustering. In *Social Informatics: 9th International Conference, SocInfo 2017, Oxford, UK, September 13-15, 2017, Proceedings, Part II 9*, pages 378–390. Springer, 2017.

[161] Li Wang, Wei Zhang, and Mei Huang. Dynamic resource allocation for large-scale distributed training of deep learning models. In *Proceedings of the ACM Symposium on Cloud Computing*, pages 789–800. ACM, 2020.

[162] Zhuang Wang, Zhen Jia, Shuai Zheng, Zhen Zhang, Xinwei Fu, T. S. Eugene Ng, and Yida Wang. Gemini: Fast failure recovery in distributed training with in-memory checkpoints. In *Proceedings of the 29th Symposium on Operating Systems Principles*, SOSP '23, page 364–381, New York, NY, USA, 2023. Association for Computing Machinery.

[163] Zihao Wang, Bin Cui, and Shaoduo Gan. Squeezeattention: 2d management of kv-cache in llm inference via layer-wise optimal budget, 2024.

[164] Wikimedia Foundation. Wikipedia database dumps. https://dumps.wikimedia.org/, 2024. Accessed: 2024-10-18.

[165] BigScience Workshop, :, Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, Jonathan Tow, Alexander M. Rush, Stella Biderman, Albert Webson, Pawan Sasanka Ammanamanchi, Thomas Wang, Benoît Sagot, Niklas Muennighoff, Albert Villanova del Moral, Olatunji Ruwase, Rachel Bawden, Stas Bekman, Angelina McMillan-Major, Iz Beltagy, Huu Nguyen, Lucile Saulnier, Samson Tan, Pedro Ortiz Suarez, Victor Sanh, Hugo Laurençon, Yacine Jernite, Julien Launay, Margaret Mitchell, Colin Raffel, Aaron Gokaslan, Adi Simhi, Aitor Soroa, Alham Fikri Aji, Amit Alfassy, Anna Rogers, Ariel Kreisberg Nitzav, Canwen Xu, Chenghao Mou, Chris Emezue, Christopher Klamm, Colin Leong, Daniel van Strien, David Ifeoluwa Adelani, Dragomir Radev, Eduardo González Ponferrada, Efrat Levkovizh, Ethan Kim, Eyal Bar Natan, Francesco De Toni, Gérard Dupont, Germán Kruszewski, Giada Pistilli, Hady Elsahar, Hamza Benyamina, Hieu Tran, Ian Yu, Idris Abdulmumin, Isaac Johnson, Itziar Gonzalez-Dios, Javier de la Rosa, Jenny Chim, Jesse Dodge, Jian Zhu, Jonathan Chang, Jörg Frohberg, Joseph Tobing, Joydeep Bhattacharjee, Khalid Almubarak, Kimbo Chen, Kyle Lo, Leandro Von Werra, Leon Weber, Long Phan, Loubna Ben allal, Ludovic Tanguy, Manan Dey, Manuel Romero Muñoz, Maraim Masoud, María Grandury, Mario Šaško, Max Huang, Maximin Coavoux, Mayank Singh, Mike Tian-Jian Jiang, Minh Chien Vu, Mohammad A. Jauhar, Mustafa Ghaleb, Nishant Subramani, Nora Kassner, Nurulaqilla Khamis, Olivier Nguyen, Omar Espejel, Ona de Gibert, Paulo Villegas, Peter Henderson, Pierre Colombo, Priscilla Amuok, Quentin Lhoest, Rheza Harliman, Rishi Bommasani, Roberto Luis López, Rui Ribeiro, Salomey Osei, Sampo Pyysalo, Sebastian Nagel, Shamik Bose, Shamsuddeen Hassan Muhammad, Shanya Sharma, Shayne Longpre, Somaieh Nikpoor, Stanislav Silberberg, Suhas Pai, Sydney Zink, Tiago Timponi Torrent, Timo Schick, Tristan Thrush, Valentin Danchev, Vassilina Nikoulina, Veronika Laippala, Violette Lepercq, Vrinda Prabhu, Zaid Alyafeai, Zeerak Talat, Arun Raja, Benjamin Heinzerling, Chenglei Si, Davut Emre Taşar, Elizabeth Salesky, Sabrina J. Mielke, Wilson Y. Lee, Abheesht Sharma, Andrea Santilli, Antoine Chaffin, Arnaud Stiegler, Debajyoti Datta, Eliza Szczechla, Gunjan Chhablani, Han Wang, Harshit Pandey, Hendrik Strobelt, Jason Alan Fries, Jos Rozen, Leo Gao, Lintang Sutawika, M Saiful Bari, Maged S. Al-shaibani, Matteo Manica, Nihal Nayak, Ryan Teehan, Samuel Albanie, Sheng Shen, Srulik Ben-David, Stephen H. Bach, Taewoon Kim, Tali Bers, Thibault Fevry, Trishala Neeraj, Urmish Thakker, Vikas Raunak, Xiangru Tang, Zheng-Xin Yong, Zhiqing Sun, Shaked Brody, Yallow Uri, Hadar Tojarieh, Adam Roberts, Hyung Won Chung, Jaesung Tae, Jason Phang, Ofir Press, Conglong Li, Deepak Narayanan, Hatim Bourfoune, Jared Casper, Jeff Rasley, Max Ryabinin, Mayank Mishra, Minjia Zhang, Mohammad Shoeybi, Myriam Peyrounette, Nicolas Patry, Nouamane Tazi, Omar Sanseviero, Patrick von Platen, Pierre Cornette, Pierre François Lavallée, Rémi Lacroix, Samyam Rajbhandari, Sanchit Gandhi, Shaden Smith, Stéphane Requena, Suraj Patil, Tim Dettmers, Ahmed Baruwa, Amanpreet Singh, Anastasia Cheveleva, Anne-Laure Ligozat, Arjun Subramonian, Aurélie Névéol, Charles Lovering, Dan Garrette, Deepak Tunuguntla, Ehud Reiter, Ekaterina Taktasheva, Ekaterina Voloshina, Eli Bogdanov, Genta Indra Winata, Hailey Schoelkopf, Jan-Christoph Kalo, Jekaterina Novikova, Jessica Zosa Forde, Jordan Clive, Jungo Kasai, Ken Kawamura, Liam Hazan, Marine Carpuat, Miruna Clinciu, Najoung Kim, Newton Cheng, Oleg Serikov, Omer Antverg, Oskar van der Wal, Rui Zhang, Ruochen Zhang, Sebastian Gehrmann, Shachar Mirkin, Shani Pais, Tatiana Shavrina, Thomas Scialom, Tian Yun, Tomasz Limisiewicz, Verena Rieser, Vitaly Protasov, Vladislav Mikhailov, Yada Pruksachatkun, Yonatan Belinkov, Zachary Bamberger, Zdeněk Kasner, Alice Rueda, Amanda Pestana, Amir Feizpour, Ammar Khan, Amy Faranak, Ana Santos, Anthony Hevia, Antigona Unldreaj, Arash Aghagol, Arezoo Abdollahi, Aycha Tammour, Azadeh HajiHosseini, Bahareh Behroozi, Benjamin Ajibade, Bharat Saxena, Carlos Muñoz Ferrandis, Daniel McDuff, Danish Contractor, David Lansky, Davis David, Douwe Kiela, Duong A. Nguyen, Edward Tan, Emi Baylor, Ezinwanne Ozoani, Fatima Mirza, Frankline Ononiwu, Habib Rezanejad, Hessie Jones, Indrani Bhattacharya, Irene Solaiman, Irina Sedenko, Isar Nejadgholi, Jesse Passmore, Josh Seltzer, Julio Bonis Sanz, Livia Dutra, Mairon Samagaio, Maraim Elbadri, Margot Mieskes, Marissa Gerchick,

Martha Akinlolu, Michael McKenna, Mike Qiu, Muhammed Ghauri, Mykola Burynok, Nafis Abrar, Nazneen Rajani, Nour Elkott, Nour Fahmy, Olanrewaju Samuel, Ran An, Rasmus Kromann, Ryan Hao, Samira Alizadeh, Sarmad Shubber, Silas Wang, Sourav Roy, Sylvain Viguier, Thanh Le, Tobi Oyebade, Trieu Le, Yoyo Yang, Zach Nguyen, Abhinav Ramesh Kashyap, Alfredo Palasciano, Alison Callahan, Anima Shukla, Antonio Miranda-Escalada, Ayush Singh, Benjamin Beilharz, Bo Wang, Caio Brito, Chenxi Zhou, Chirag Jain, Chuxin Xu, Clémentine Fourrier, Daniel León Periñán, Daniel Molano, Dian Yu, Enrique Manjavacas, Fabio Barth, Florian Fuhrimann, Gabriel Altay, Giyaseddin Bayrak, Gully Burns, Helena U. Vrabec, Imane Bello, Ishani Dash, Jihyun Kang, John Giorgi, Jonas Golde, Jose David Posada, Karthik Rangasai Sivaraman, Lokesh Bulchandani, Lu Liu, Luisa Shinzato, Madeleine Hahn de Bykhovetz, Maiko Takeuchi, Marc Pàmies, Maria A Castillo, Marianna Nezhurina, Mario Sänger, Matthias Samwald, Michael Cullan, Michael Weinberg, Michiel De Wolf, Mina Mihaljcic, Minna Liu, Moritz Freidank, Myungsun Kang, Natasha Seelam, Nathan Dahlberg, Nicholas Michio Broad, Nikolaus Muellner, Pascale Fung, Patrick Haller, Ramya Chandrasekhar, Renata Eisenberg, Robert Martin, Rodrigo Canalli, Rosaline Su, Ruisi Su, Samuel Cahyawijaya, Samuele Garda, Shlok S Deshmukh, Shubhanshu Mishra, Sid Kiblawi, Simon Ott, Sinee Sang-aroonsiri, Srishti Kumar, Stefan Schweter, Sushil Bharati, Tanmay Laud, Théo Gigant, Tomoya Kainuma, Wojciech Kusa, Yanis Labrak, Yash Shailesh Bajaj, Yash Venkatraman, Yifan Xu, Yingxin Xu, Yu Xu, Zhe Tan, Zhongli Xie, Zifan Ye, Mathilde Bras, Younes Belkada, and Thomas Wolf. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[166] Xun Wu, Shaohan Huang, and Furu Wei. Mixture of lora experts. *arXiv preprint arXiv:2404.13628*, 2024.

[167] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 38087–38099. PMLR, 23–29 Jul 2023.

[168] Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv*, 2023.

[169] Zhen Xie, Murali Emani, Xiaodong Yu, Dingwen Tao, Xin He, Pengfei Su, Keren Zhou, and Venkatram Vishwanath. Centimani: Enabling fast AI accelerator selection for DNN training with a novel performance predictor. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 1203–1221, Santa Clara, CA, July 2024. USENIX Association.

[170] Wei Xu, E-Sheng Peh, and Edward Wong. Fusing kernels for higher performance deep learning. In *Proceedings of the 2017 ACM SIGPLAN International Conference on Compiler Construction*, pages 1–12. ACM, 2017.

[171] Dongjie Yang, XiaoDong Han, Yan Gao, Yao Hu, Shilin Zhang, and Hai Zhao. Pyramidinfer: Pyramid kv cache compression for high-throughput llm inference, 2024.

[172] June Yong Yang, Byeongwook Kim, Jeongin Bae, Beomseok Kwon, Gunho Park, Eunho Yang, Se Jung Kwon, and Dongsoo Lee. No token left behind: Reliable kv cache compression via importance-aware mixed precision quantization, 2024.

[173] Yifan Yang, Joel S Emer, and Daniel Sanchez. Trapezoid: A versatile accelerator for dense and sparse matrix multiplications. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 931–945. IEEE, 2024.

[174] Ziyu Ying, Sandeepa Bhuyan, Yan Kang, Yingtian Zhang, Mahmut T. Kandemir, and Chita R. Das. Edgepc: Efficient deep learning analytics for point clouds on edge devices. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, New York, NY, USA, 2023. Association for Computing Machinery.

[175] Ziyu Ying, Shulin Zhao, Haibo Zhang, Cyan Subhra Mishra, Sandeepa Bhuyan, Mahmut T. Kandemir, Anand Sivasubramaniam, and Chita R. Das. Exploiting frame similarity for efficient inference on edge devices. In *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, pages 1073–1084, 2022.

[176] Huizi Yu, Lizhou Fan, Lingyao Li, Jiayan Zhou, Zihui Ma, Lu Xian, Wenyue Hua, Sijia He, Mingyu Jin, Yongfeng Zhang, Ashvin Gandhi, and Xin Ma. Large language models in biomedical and health informatics: A review with bibliometric analysis. *arXiv preprint arXiv:2403.16303*, 2024.

[177] Ann Yuan, Andy Coenen, Emily Reif, and Daphne Ippolito. Wordcraft: story writing with large language models. In *Proceedings of the 27th International Conference on Intelligent User Interfaces*, pages 841–852, 2022.

[178] Ted Zadouri, Ahmet Üstün, Arash Ahmadian, Beyza Ermiş, Acyr Locatelli, and Sara Hooker. Pushing mixture of experts to the limit: Extremely parameter efficient moe for instruction tuning. *arXiv preprint arXiv:2309.05444*, 2023.

[179] Haibo Zhang, Shulin Zhao, Ashutosh Pattnaik, Mahmut T. Kandemir, Anand Sivasubramaniam, and Chita R. Das. Distilling the essence of raw video to reduce memory usage and energy at edge devices. In *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '52, page 657–669, New York, NY, USA, 2019. Association for Computing Machinery.

[180] Hengrui Zhang, August Ning, Rohan Baskar Prabhakar, and David Wentzlaff. Llmcompass: Enabling efficient hardware design for large language model inference. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 1080–1096, 2024.

[181] Nan Zhang, Yanchi Liu, Xujiang Zhao, Wei Cheng, Runxue Bao, Rui Zhang, Prasenjit Mitra, and Haifeng Chen. Pruning as a domain-specific LLM extractor. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1417–1428, Mexico City, Mexico, June 2024. Association for Computational Linguistics.

[182] Wei Zhang, Chen Liu, Xinyu Wang, and Jian Li. Efficient load balancing for distributed training of deep neural networks. In *Proceedings of the IEEE International Conference on Big Data*, pages 1234–1243. IEEE, 2019.

[183] Yuanrui Zhang, Wei Ding, Mahmut Kandemir, Jun Liu, and Ohyoung Jang. A data layout optimization framework for nuca-based multicores. In *Proceedings of the 44th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 489–500, 2011.

[184] Yuanrui Zhang, Wei Ding, Jun Liu, and Mahmut Kandemir. Optimizing data layouts for parallel computation on multicores. In *2011 International Conference on Parallel Architectures and Compilation Techniques*, pages 143–154. IEEE, 2011.

[185] Yuanrui Zhang, Mahmut Kandemir, and Taylan Yemliha. Studying inter-core data reuse in multicores. *ACM SIGMETRICS Performance Evaluation Review*, 39(1):25–36, 2011.

[186] Yusen Zhang, Ansong Ni, Ziming Mao, Chen Henry Wu, Chenguang Zhu, Budhaditya Deb, Ahmed Awadallah, Dragomir Radev, and Rui Zhang. Summ^n: A multi-stage summarization framework for long input dialogues and documents. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1592–1604, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[187] Yusen Zhang, Ruoxi Sun, Yanfei Chen, Tomas Pfister, Rui Zhang, and Sercan Ö Arik. Chain of agents: Large language models collaborating on long-context tasks. In *Advances in Neural Information Processing Systems*, 2024.

[188] Zhenyu Zhang, Shiwei Liu, Runjin Chen, Bhavya Kailkhura, Beidi Chen, and Atlas Wang. Q-hitter: A better token oracle for efficient llm inference via sparse-quantized kv cache. In P. Gibbons, G. Pekhimenko, and C. De Sa, editors, *Proceedings of Machine Learning and Systems*, volume 6, pages 381–394, 2024.

[189] Zhenyu Zhang, Ying Sheng, Tianyi Zhou, Tianlong Chen, Lianmin Zheng, Ruisi Cai, Zhao Song, Yuandong Tian, Christopher Ré, Clark Barrett, Zhangyang "Atlas" Wang, and Beidi Chen. H2o: Heavy-hitter oracle for efficient generative inference of large language models. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 34661–34710. Curran Associates, Inc., 2023.

[190] Shulin Zhao, Prasanna Venkatesh Rengasamy, Haibo Zhang, Sandeepa Bhuyan, Nachiappan Chidambaram Nachiappan, Anand Sivasubramaniam, Mahmut Taylan Kandemir, and Chita Das. Understanding energy efficiency in iot app executions. In *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pages 742–755, 2019.

[191] Tianyu Zhao, Sharan Narang, Kelvin Guu, Angela Fan, Oriol Vinyals, William W Cohen, and Wei Lu. Measuring massive multitask language understanding. *arXiv preprint arXiv:2110.11605*, 2021.

[192] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[193] Youpeng Zhao, Di Wu, and Jun Wang. Alisa: Accelerating large language model inference via sparsity-aware kv caching. In *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pages 1005–1017, Los Alamitos, CA, USA, Jul 2024. IEEE Computer Society.

[194] Zexuan Zhong, Mengzhou Xia, Danqi Chen, and Mike Lewis. Lory: Fully Differentiable Mixture-of-Experts for Autoregressive Language Model Pre-training. *arXiv preprint arXiv:2405.03133*, 2024.

[195] Zeyang Zhong, Urvashi Khandelwal, Omer Levy, and Dan Jurafsky. Beyond common sense: The story of hellaswag. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4599–4604. Association for Computational Linguistics, 2020.

[196] Yanqi Zhou, Nan Du, Yanping Huang, Daiyi Peng, Chang Lan, Da Huang, Siamak Shakeri, David So, Andrew Dai, Yifeng Lu, Zhifeng Chen, Quoc Le, Claire Cui, James Laudon, and Jeff Dean. Brainformers: Trading simplicity for efficiency. pages 42531–42542. PMLR, 2023.

[197] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, and James Laudon. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.

[198] Michael Zhu and Suyog Gupta. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*, 2017.

[199] Yang Zhu, Luke Zettlemoyer, and Jimmy Ba. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 58–68. Association for Computational Linguistics, 2015.

[200] Yazhou Zu, Alireza Ghaffarkhah, Hoang-Vu Dang, Brian Towles, Steven Hand, Safeen Huda, Adekunle Bello, Alexander Kolbasov, Arash Rezaei, Dayou Du, Steve Lacy, Hang Wang, Aaron Wisner, Chris Lewis, and Henri Bahini. Resiliency at scale: Managing {Google's}{TPUv4} machine learning supercomputer. In *21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24)*, pages 761–774, 2024.

Data Management and Sharing Plan

This section describes "data" – in its most general definition that includes, compiler, scheduler, and simulation source codes and executables, LLM/expert models/algorithms, educational materials, benchmarks, and experimental data – that will be produced during the project; how this data will be managed, stored and shared, what standards will be used for different types of data, and how data will be handled and protected during and after the project. It also describes our plans for ensuring data integrity and reproducibility.

1. Types of Data and Storage

The project will generate seven types of data: (i) the codes and executables for the compiler that performs expert-to-chiplet mapping; (ii) source codes for scheduling support and simulator; (iii) expert repository that will hold the LLMs/expert models generated during the project; (iv) detailed LLM/expert algorithms as well as workload characterization and experimental data; (v) educational materials; (vi) a document detailing how to use the software developed during the project; and (vii) finally, lineage (provenance) data (more on this below). The source codes for the compiler and systems software support, as well as simulator source code will be maintained in Penn State as long as they are needed. They will be made available to the broad research community and other interested parties via a GitHub license. The educational materials, characterization and experimental data, and the representative LLM/expert models will also be maintained in machines at Penn State, and will be shared with the user community via a website dedicated to the project (as discussed earlier).

2. Data and Metadata Standards

The workload characterization and experimental data will be compressed and made available to interested parties in a compressed format. The compiler and system software code and other design artifacts will be maintained in both source formats (e.g., C/C++/C#/Python files) as well as in binary, in an open-source fashion. The educational material will be stored in text, MSWORD, PowerPoint, PDF, and various video formats. When needed, this material will be ported to other formats as well. Metadata will most likely be needed as a part of curriculum development process and training. These will be stored in XML and web formats. The lineage data will use the formats required by the underlying data lineage tools.

3. Access and Sharing Policies

We will publish our findings from this research in top ML/AI, systems, HPC, computer architecture and performance evaluation venues, including, but not limited to, ICML, ICLR, NeurIPS, MLSys, ACM SC, ISCA, MICRO, HPCA, ASPLOS and SIGMETRICS, as well as the top relevant ACM and IEEE journals and transactions. When appropriate, the PIs will also try to share their research findings with the broad research community via posts, talks, and seminars. No ethical and privacy issues will be associated with the data, and the data will not contain any personal information and will not be copyrighted. All data mentioned above will be accessible via our website dedicated to the project.

4. Policies and Provisions for Reuse and Redistribution

No permission restrictions will be placed on the data. Academic researchers, industrial researchers as well as scientists working on AI/ML (especially generative AI and LLMs), systems, computer architecture, carbon-efficient cyberinfrastructure, compilers and performance evaluation areas would likely be interested in our data. Furthermore, no restriction on the reuse or redistribution of any artifact developed in this project will be placed for non-commercial use. Also, we will prepare educational materials – in a slide-deck form – that can be easily used in different classes like ML and systems.

5. Archiving and Preservation Policies

Data will be archived on our departmental machines at Penn State as long as necessary for possible academic publications and at least until the end of the project. The main documentation that will accom-

pany the data are project reports and research publications. Since all data will be maintained in electronic format, archiving and version control will be achieved automatically via SVN software [15]. The lecture slides/videos are expected to be stored until their useful lifetime, and they will also be made available via YouTube. Finally, the curriculum materials will be stored as long as they are used to enhance the courses that PIs teach at Penn State. We will also carve up short lecture materials based on the project and preserve them on departmental machines. Note that the lineage data/metadata will be updated as more characterization and experimental results as well as models are collected/generated or the compiler/system software source codes are updated.

6. Data Lineage and Reproducibility

All three PIs are fully committed to "reproducibility" and open access policy. In addition to the data described above, they will also generate and maintain "annotations" attached to i) the libraries used in compiler and runtime system source codes, and ii) the characterization and experimental data generated by the project. These annotations will provide a kind of "data lineage", i.e., they will document a trail that accounts for the origin of a piece of data as well as the stages it went through to reach its final form. In a sense, our annotations will make the data originating from this project more actionable and easily reproducible. To generate such annotations, where appropriate, we plan to use well-established data lineage tools such as Keboola [78] and Octopai [120]. The data lineage information will also help the project to reduce its "storage footprint". More specifically, instead of storing all versions of each and every dataset and model we generate, we can only store the most important ones and, for the remaining ones, the data lineage information (a kind "metadata") can be used to re-generate them, if/when needed.

Collaboration Plan

Project Team

The proposed project spans design and analysis of LLM and expert models, characterization and evaluation of such models, development of compiler and runtime system support for efficient LLM/expert training and inference as well as chiplet selection for LLM/expert execution. The project will be managed by the three PIs from Penn State. The specific responsibilities of the PIs and their complementary expertise are explained below:

Chitaranjan Das (PI): Das is the PI of the project and will be responsible for the overall coordination and progress as planned in the project schedule. His expertise includes multicore architectures, architectural optimization of ML kernels, on-chip and chip-to-chip interconnect design, cloud computing, and performance evaluation. He will lead Thrust-3 and also co-lead Thrust-4 with Co-PIs Zhang and Kandemir.

Mahmut Taylan Kandemir (Co-PI): Kandemir's expertise includes optimizing compilers, storage systems, HPC, and workload characterization. He will lead Thrust-2 and collaborate with Das and Zhang in Thrust-4.

Rui Zhang (Co-PI): Zhang's research expertise includes LLMs, trustworthy human-centered AI, and AI for science. He has an extensive research background and publication record in efficient methods for LLMs such as LLM pruning (NAACL 2024), LLM parameter-efficient finetuning (ACL 2022, EMNLP 2023), long-context LLMs (ACL 2022, NeurIPS 2024), data selection for LLM in-context learning (ICLR 2023). In the context of this project, he will lead Thrust-1 and also co-lead Thrust-4 with Kandemir and Das.

All the three PIs will work in close coordination on the individual research topics as well as the overall integration of the project. The expertise of the management team members are complementary, and collectively cover all major aspects of the proposed research, namely, algorithm design, system-level support and architectural support. Das and Kandemir have worked together in prior and on-going NSF projects and have a history of successful collaboration, including co-advising underrepresented students. Zhang, Kandemir and Das have recently started to work together. Zhang and Kandemir have recently co-authored a paper in MICRO (2024) and all three PIs have co-authored a paper in NeurIPS (2021).

Student Support: The project will support four PhD students for the proposed three-year duration of the project. The students will work on separate thrusts in the beginning (one student will be the primary contact for each thrust), but they will works together in the last year for integrating the different components of the research for a comprehensive evaluation and refinements of the proposed models, algorithms, compiler and system support. In addition, we will also include undergraduates from the Schreyer Honors College at Penn State, and specifically draw undergraduate students from underrepresented groups, who are interested in pursuing graduate studies. We will seek REU supplements to support the undergraduate students for working on this project. The investigators have a very good track record of advising undergraduate students (resulting in more than 20 undergraduate honors thesis), and they will continue to do so in this project as well.

Project Timeline

Figure 8 despicts a tentative projected timeline for the proposed work. While, we will start all four thrusts in year 1, by the middle of the second year, we expect to have our initial expert models and system support to be in place. By the end of the second year, the preliminary system support will be finalized and we will have the initial expert-to-chiplet mappings ready. Around mid-way in the third year, we will have all three main pieces of the project (algorithmic enhancements, system support and architectural support) ready. By the end of the project, the entire framework along with sample expert models, algorithms, system support and documentation will be in the public domain.

Our simulation testbed will be tested, refined if necessary, and will be updated in the public domain (Github). The results will be disseminated through timely scientific publications in respected conferences and journals throughout the project period. Note that the education activities, undergraduate involvement efforts, outreach and BPC activities, and industry collaboration efforts (discussed earlier in the broader impact section of the proposal) will continue throughout the entire project duration.

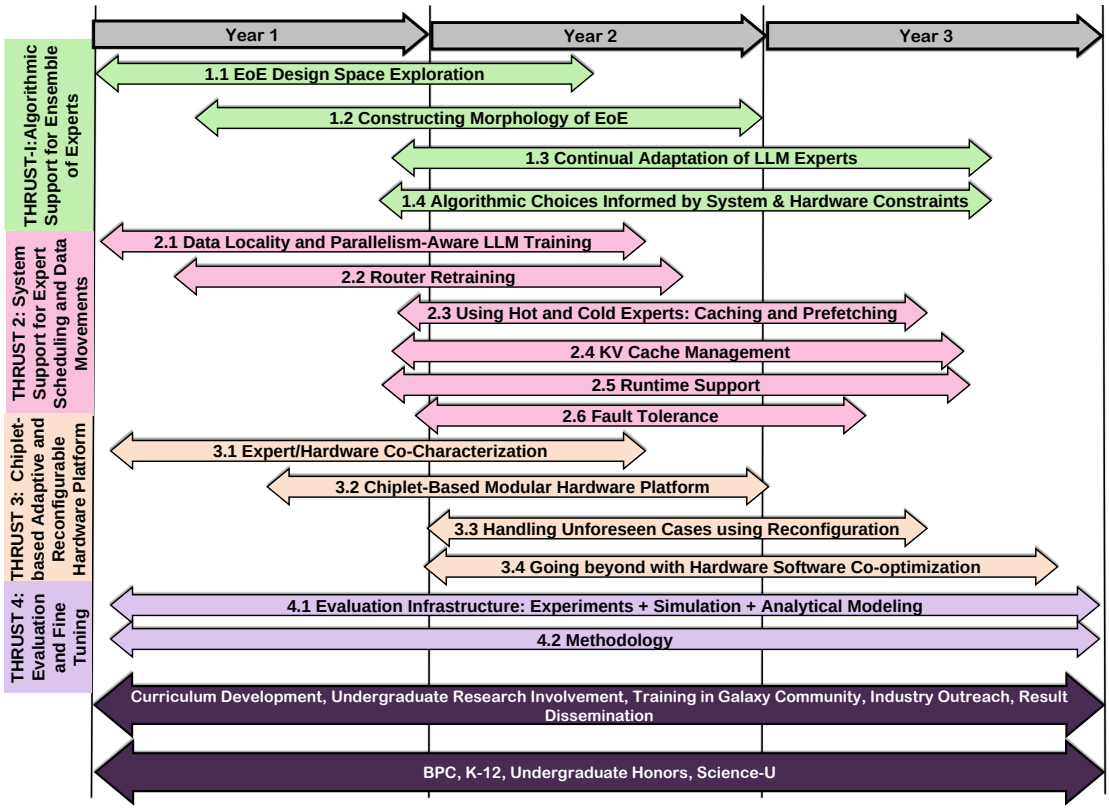

Figure 8*: Project timeline that shows both research and educational/outreach efforts.*

Collaboration Logistic

The PIs will meet regularly to assess the overall progress of the project. They will not only evaluate whether the individual and intermediate research goals have been achieved or not, but *will also discuss the progress towards achieving the educational and outreach/BPC goals of the project.* The necessary adjustments will be made if the observed progress is not satisfactory. The PIs will meet with the PhD students as frequently as needed. The existing industrial partnerships of the PIs (e.g., with Intel, AMD, NVIDIA, Microsoft, and Amazon) will be leveraged to provide opportunities for student internships and placements. Moreover, as stated in the proposal, we will collaborate with Argonne National Lab specifically for the experimental evaluation thrust of the project. A letter of collaboration is included as a supplementary document (our partnership with ANL already generated a SIGMETRICS paper [26], and we will also be submitting a joint paper to MLSys by the end of October 2024). These partnerships will also provide external insights to the project, and in particular, our collaboration with ANL enable us to access a large number and variety of compute platforms (including those with accelerators).

Project Website

A website will be maintained for the project. This website will have both "external" and "internal" interfaces. The external interface will contain links to the publicly-available LLM/expert models and algorithms, compiler, system software and simulator codes and experimental data as they become available, whereas the internal link will be used to facilitate code sharing among the students and the PIs, as well as tracking the progress of code development and publication-related efforts.

Broadening Participation in Computing (BPC) Plan: Connected

Since this is a connected BPC plan, we only discuss the planned activities for the PIs, as specified in the submission guidelines. The approved departmental BPC plan is also included.

Plan of Activities: Aligned with the departmental BPC plan, we plan to pursue several activities related to this proposed research as summarized below:

- **Customized Graduate Student Recruiting and Training:** We will recruit graduate women and students from populations underrepresented in computing to work on this project. For this, we will collaborate with the Multicultural Engineering Graduate Student Association (MEGA) and Graduate Women in Engineering (GradWIE) programs. Moreover, the CoE (College of Engineering) is a partner with the National GEM Consortium that leads the Grad Lab, which facilitates the participation of populations underrepresented in computing for graduate studies in engineering and science. In addition to working with the GEM/Grad lab, department has developed a partnership with the Black in AI (BAI) group to attract more students from populations underrepresented in computing to our graduate program. We also plan develop recruiting relationships with HBCUs. All these efforts should help us in broadening participation in this project. Aligned with the departmental BPC plan, the students working on this project will participate in various diversity, equity, inclusion and belonging (DEIB) DEIB activities.
- **Summer Camp for Middle School Girls:** PI Das has been involved with organization of week-long summer camps (funded by the CSE Department) targeted at middle school girls since 2017. Our earlier camps have already introduced participants to basic concepts in programming, building vision systems that assist visually impaired and learning skills towards building a basic embedded vision system, programming for robotics and exposure to various emerging tools in computing. We will continue to develop new week-long courses introducing students to LLMs/Generative AI and inspiring them through hands-on application and system building activities. PI Zhang has participated in the 2024 summer camp, and he will lead this effort for the future years. This summer camp has been very successful, and we plan to extend it to additional school districts.
- **Summer Research Opportunities for High School Students and Science Teachers:** The PIs has been participating in the organization of various summer activities with high school students and teachers. For example, PI Kandemir has previously participated in the organization of a workshop targeting high school teachers and gave a talk on ML and high-performance computing. The research team is planning to organize summer activities with high school students and teachers in the context of this project as well. More specifically, to complement the summer camp with a more in-depth exploration for the students, we will work with the local school science teachers and students to develop the inquisitiveness of the potential of LLM concepts and develop simple but interesting LLM-based applications to attract young students. For example, PI Zhang will lead a workshop for high school students featured with an LLM and society seminar and a Computational Linguistics Olympiad competition to inspire students' interests in CS and AI. Additionally, PI Kandemir will organize an LLM workshop targeting high school teachers, discussing topics such as capabilities of LLMs, how their power and limits can be explained to students, and ethical considerations in using the LLM-based tools. The PIs will seek funding for these activities from Penn State, and in particular from the ICDS (Institute for Computational and Data Sciences).
- **Undergraduate Research Experience:** The PIs have participated in the Summer Research Opportunities internship program that hosts students from under-represented communities with interest in pursuing graduate studies. PI Das has been involved with organizing the *Visit In Engineering Weekend* (VIEW) program for students entering their junior and senior years of high school, which fosters interest in engineering. Participants carry out hands-on design activities with faculty and students. We plan to provide hands-on computer architecture experience to students and show them how modern computer systems can address important societal challenges, specifically how LLMs can be used in many such domains. PI Zhang has offered an NSF REU seminar and mentored an NSF REU student on a project investigating the intersection of LLMs and code generation. The PIs will leverage their complementary experience in developing appropriate research thrust areas within the scope of LLM to attract and engage a new cohort set of minority undergraduate students in research.
- **Broad Reach to K-12 students:** We will partner with CSATS (Center for Science and the Schools) in the College of Education at Penn State to leverage their ongoing Penn State STEM-oriented outreach programs. We have had continuous partnership with the CSATS faculty for over 5 years and have hosted

day-long seminars for middle school and high school teachers as part of our prior NSF funded outreach efforts. In particular, we plan to visit along with a couple of our trained undergraduate and graduate students to local middle and high schools and talk to students about the exciting opportunities in computing discipline. A selected group of undergraduate students and volunteers from CRA-W, ACM, and Girls Who Code programs will be trained to go to these schools and talk to students. In this context, we plan to show them how Generative AI are being used in different application domains (e.g., LLM-powered story generation and creative writing, interactive chatbots for math learning and problem solving, etc).
We have also participated in the annual Exploration-U Science day events, organized by Penn State. This event helps to create awareness to a much broader audience to our targeted efforts for summer camps and summer internship opportunities. It also provides us access to new contacts and additional recruitment opportunities of diverse students. For example, we have performed demonstration events at the local Science Museum for kids and the regional arts festival based on interaction with individuals who visited our exhibits. The PIs and their students are passionate about the broader outreach and in kindling interest in the K-12 students to pursue STEM careers. In the context of this project, we plan to participate in Exploration-U Science Day and organize a booth on Generative AI.

Preparation for Activities: The two senior PIs have extensive prior experience in supervising female undergraduate and graduate students. Both of them have graduated 15+ female PhD and MS students (5 in last five years) and a few of them have taken up faculty positions at different schools. They have also advised a couple of female undergraduate students. PI Das has worked with several high school students and teachers in summer for the completed NSF Expeditions project. In addition, he had co-organized a summer workshop for visually-impaired students as a part of their Expeditions project. As the department Head, PI Das has been closely associated with many such activities. PI Kandemir is an adviser/co-adviser of 5 female students, and PI Zhang has been advising 1 female PhD student, 4 female undergraduate students for their honors thesis projects, and serving on PhD dissertation committee for 9 female PhD students including 1 African American student.

We plan to share our BPC effort outcomes in our NSF project reports and at different forums such as the annual Big Ten Department Heads meeting as well as in Tapia and CRA-W conferences.

Mentoring Plan

This project will accommodate a total of 4 PhD students, as discussed in our Management and Coordination Plan. The PIs will perform the following mentoring activities for these PhD students:

Orientation and Expectation Setting

The PIs will engage in in-depth conversations with PhD students, to set clear expectations, goals, and deliverables for the project period. After the initial meetings with the PhD students, the students will be asked to complete a worksheet to ensure alignment on objectives. Regular annual review meetings will be conducted to assess progress and make necessary adjustments.

Career Counseling/Advising

The PIs will provide – on a regular basis – career counseling and advising to the PhD students in the project as part of the mentorship. The students will also have access to individual career counseling appointments with Penn State, who specialize in career and professional development. Additionally, the students will be encouraged to attend career and professional development workshops offered by Penn State.

Training in Paper Writing

The PIs will discuss regularly with the PhD students the best paper-writing practices, to ensure that they gain the first-hand experience in best practices. To expedite the process, where it makes sense, the PIs will team up the new students with the older ones in paper writing process.

Publications and Presentations

The PhD students will receive guidance and training in the preparation of manuscripts for scientific journals and presentations at conferences and workshops. They will have access to courses on Effective Communication and Presentation Skills.

Improving Skills

The students will participate in regular research group meetings, where they will describe their work to colleagues and collaborate on solving research problems, fostering communication, programming skills, and other types of technical skills.

Instruction in Responsible Professional Practices

The PhD students will receive instruction in responsible and ethical professional practices regularly within the context of their work. Training will cover the fundamentals of the scientific method, data protection and ethical sharing, lab safety, and other standards of professional practice. They will also be encouraged to affiliate with one or more professional societies in their chosen field. They will also have access to the various Responsible Conduct of Research and DEIB (Diversity, Equity, Inclusion, and Belonging) training courses offered by Penn State.

Diverse Collaborations

Finally, the PhD students in the project will be encouraged to engage in collaborations with researchers from diverse backgrounds and disciplinary areas to enhance their collaboration and communication skills.

Facilities, Equipment, and Other Resources

The participants of the proposed project have access to the facilities and resources described below.

Lab Resources: The PIs collectively have more than 6,000 sq. ft. of laboratory space including three conference rooms and more than 60 desks with PCs, peripherals, and virtual meeting equipment. The prototyping efforts in the proposed research will be carried out primarily in the research labs at Penn State, directed by Das, Kandemir, and Zhang.

The PIs' labs house several medium-sized clusters of rack-mounted servers connected via 10Gbps Ethernet switches. A large portion of our test and development will be performed on these machines. The PIs and their students also have access to grid and supercomputing resources through the College of Engineering at University Park. A variety of other grid computing resources are also freely available to their research groups. For example, Kandemir's lab has CAPI-capable FPGA resources integrated into IBM Power servers for modeling of emerging accelerator and chiplet resources and software licenses for the HLS and EDA flows needed to generate new accelerator and chiplet models and deploy them on this platform.

CSE Department Resources: The Department of Computer Science and Engineering (CSE) at Penn State uses a network of Linux, OS X, and Windows workstations and servers to support academic computing needs. Instruction is supported by highly virtualized services providing file, application, and license servers for approximately 350 workstations in labs, graduate student offices, and faculty. Six student teaching labs are equipped to host digital design, FPGA, circuit design, programming, robotics/drone, and related curricula.

Funded research efforts support 5 High-Performance Computing Clusters, totaling over 300 compute nodes sharing IBA, Myrinet, and GigE interconnection. The clusters offer researchers HPC and GPU configuration 9both A 100 and H 100)agility to target highly specialized use-cases. Researchers currently share approximately 105TB of NAS storage. The department operates its own firewall infrastructure using Palo Alto, Cisco, and Sonicwall products.

The CSE department at Penn State currently maintains copies of the LLVM compiler toolset, various LLM models, as well as various ML packages and HPC libraries.

Other Resources: Penn State Institute for Computational and Data Sciences (ICDS), of which Mahmut Kandemir is an associate director, provides a variety of compute, storage and network resources, various IT services, including operations, backup, technical consulting, and training material, and is compliant with specific NSF, NIH, and NIST security controls. It offers over 1000 servers with over 40,000 processing cores, over 300 NVIDIA GPUs, 5 Petabytes (PB) of disk parallel file storage and 10 PB of archive storage, high-speed Ethernet and Infiniband interconnects, and a large software stack. The PIs also have access to NSF CloudBank, ACCESS and various NERSC resources.

Also, Kandemir was a co-PI on a recent MRI award. This grant enables the PIs as well as the project team to access, among other resources, hundreds of Intel Xeon nodes, various types of NVIDIA GPUs (A100 and V100), and two large storage arrays consisting of various types of HDDs, SSDs, FPGAs, as well as 4 computational storage devices (Samsung SmartSSD). The PIs will also have access to the Argonne National Laboratory Aurora Exascale Supercomputer – a collection of LLM accelerators for our testbed and experimental evaluations.

Finally, our partnership with ANL (see the attached collaboration letter) allows us to access ANL resources, including hardware accelerators for LLM.

Office Space: Each PI has an office that is approximately 75 sq. ft., including desks with workstations and peripherals.

List of Project Personnel and Partner Institutions

1. Chitaranjan Das; Pennsylvania State University; PI
2. Mahmut Taylan Kandemir; Pennsylvania State University; co-PI
3. Rui Zhang; Pennsylvania State University; Co-PI
4. Murali Emani; Argonne National Laboratory; Unfunded Collaborator