

Evaluation of pre-training large language models on leadership-class supercomputers

This Accepted Manuscript (AM) is a PDF file of the manuscript accepted for publication after peer review, when applicable, but does not reflect post-acceptance improvements, or any corrections. Use of this AM is subject to the publisher's embargo period and AM terms of use. Under no circumstances may this AM be shared or distributed under a Creative Commons or other form of open access license, nor may it be reformatted or enhanced, whether by the Author or third parties. By using this AM (for example, by accessing or downloading) you agree to abide by Springer Nature's terms of use for AM versions of subscription articles: <https://www.springernature.com/gp/open-research/policies/accepted-manuscript-terms>

The Version of Record (VOR) of this article, as published and maintained by the publisher, is available online at: <https://doi.org/10.1007/s11227-023-05479-7>. The VOR is the version of the article after copy-editing and typesetting, and connected to open research data, open protocols, and open code where available. Any supplementary information can be found on the journal website, connected to the VOR.

For research integrity purposes it is best practice to cite the published Version of Record (VOR), where available (for example, see ICMJE's guidelines on overlapping publications). Where users do not have access to the VOR, any citation must clearly indicate that the reference is to an Accepted Manuscript (AM) version.

Evaluation of Pre-Training Large Language Models on Leadership-class Supercomputers *

Junqi Yin, Sajal Dash, John Gounley, Feiyi Wang
and Georgia Tourassi

National Center for Computational Sciences, Oak Ridge
National Laboratory, Oak Ridge, TN, USA.

*Corresponding author(s). E-mail(s): yinj@ornl.gov;

Contributing authors: dashs@ornl.gov; gounleyjp@ornl.gov;
fwang2@ornl.gov; tourassig@ornl.gov;

Abstract

Large language models (LLMs) have arisen rapidly to the center stage of artificial intelligence as the foundation models applicable to many downstream learning tasks. However, how to effectively build, train, and serve such models for many high-stake and first-principle based scientific use cases are both of great interests and of great challenges. Moreover, pre-training LLMs with billions or even trillions of parameters can be prohibitively expensive not just for academic institutions, but also for well-funded industrial and government labs. Furthermore, the energy cost and the environmental impact of developing LLMs must be kept in mind. In this work, we conduct a first-of-its-kind performance analysis to understand the time and energy cost of pre-training LLMs on the Department of Energy (DOE)'s leadership-class supercomputers. Employing state-of-the-art distributed training techniques, we evaluate the computational performance of various parallelization approaches at scale for a range of model sizes, and establish a projection model for the cost of full training. Our findings provide baseline

*This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The publisher, by accepting the article for publication, acknowledges that the U.S. Government retains a non-exclusive, paid up, irrevocable, world-wide license to publish or reproduce the published form of the manuscript, or allow others to do so, for U.S. Government purposes. The DOE will provide public access to these results in accordance with the DOE Public Access Plan (<http://energy.gov/downloads/doe-public-access-plan>).

2 *Pre-Training Large Language Models*

results, best practices, and heuristics for pre-training such large models that should be valuable to HPC community at large. We also offer insights and optimization strategies for using the first exascale computing system, Frontier, to train models of the size of GPT-3 and beyond.

Keywords: AI foundation model, distributed training, performance analysis

1 Introduction

It has been well recognized that the convolution neural network architecture led to the first wave of AI breakthroughs, beginning around 2010. With the introduction of the Transformer model [1], especially large language models (LLMs) such as Bidirectional Encoder Representations from Transformers (BERT) [2], Generative Pre-trained Transformer (GPT) [3], a new wave of innovations is sweeping across various fields. A new paradigm [4] for building AI systems has been introduced, in which a ‘foundation model’ trained on large unlabelled data can serve as a base for many downstream tasks. Notable examples include OpenAI’s GPT3 [5], GPT4 [6] and DALL-E [7], Meta’s OPT [8] and LLaMA [9], and BigScience’s BLOOM [10], to name but a few. All these efforts stem from commercial side, and a large gap is yet to be filled for scientific applications.

There are early efforts in applying language models to scientific texts. Several models have been introduced, including SciBERT [11], MatBERT [12], ScholarBERT [13], and a GPT-style model [14]. The model sizes are as large as 1.47B parameters. Considering the overall volume of the scientific literature (e.g., the CORE [15] dataset alone contains 207M papers, over 200B tokens), a LLM of 10B parameters or more can be built based on the compute-optimal training recommendation, i.e. to maintain a 20 to 1 ratio of tokens vs. number of parameters [16] (although recent LLaMA models [9] indicate a much higher tokens-to-parameter ratio). Other potential applications for LLMs are scientific sequence or imaging data, both of which also involve large datasets.

However, pre-training LLMs is prohibitively costly even for large institutions. It has been reported [17] that training a GPT3 model from scratch will cost over 1 million dollars on AWS in the best case scenario where no hyper-parameter tuning (e.g., batch size, learning rate, etc.) is necessary. In typical cases, it would require extensive hyper-parameter tuning to obtain the best model. Furthermore, LLMs often can not fit into a single device due to the memory limitation, and advanced parallelization techniques are required, which involves additional tuning for the underlying hardware and software to achieve the best computational performance.

To understand the cost of building foundation models and establish best practices for training LLMs, we evaluate two distributed training frameworks on two supercomputers with different accelerators and software stacks: the Nvidia V100-based Summit and AMD MI250X-based Crusher. We compare

different parallel strategies for various LLMs and project the full training cost in terms of both compute hours and energy. Our contributions include:

- Establishing performance baselines by benchmarking two LLM-training frameworks on both Nvidia- and AMD-based supercomputers.
- Developing cost models for runtime and energy projections.
- Drawing insights on future platforms for training LLMs.

The rest of the paper is organized as follows: Section 2 provides a general background on the LLM architecture and the scaling law. Section 3 details the distributed training approach and our analysis methods. Section 4 presents our results on performance evaluations and cost analysis, and discusses the gap for current practices and implications for future systems. We conclude in Section 5.

2 Background and related work

In this paper, we focus on specifically Transformer-based LLMs, which are state-of-the-art for many natural language processing [5] and computer vision [18] tasks. We now give a brief overview on the architectures of the Transformer, BERT, and GPT, as well as the important scaling laws which describe the heuristic and experimental relationships between model accuracy, efficiency, model size (number of model parameters), data size (number of tokens), and compute cycles (number of floating-point operations).

Transformer Architecture

A Transformer is a self-attention based deep learning model for analyzing sequential data without requiring recurrence or convolution [1]. This model has two major components: an encoder and a decoder (Figure 1). This architecture processes the entire sequence at once (unlike a recurrence model) but uses positional embedding to delineate between the relative positions of input tokens. In the encoder, there are identical layers of multi-head self-attention followed by a feed-forward network that learns context around a token and creates a continuous representation. The decoder has a similar structure, and takes in the output from the encoder and the decoder output from earlier step to predict the outcome.

BERT Architecture

Bidirectional Encoder Representations from Transformers (BERT) [2] is a language model consisting of the encoder part of the Transformer model. This model allows pre-training of deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers (Figure 2, left). Additional output layers are used for fine-tuning the pre-trained model for a variety of tasks.

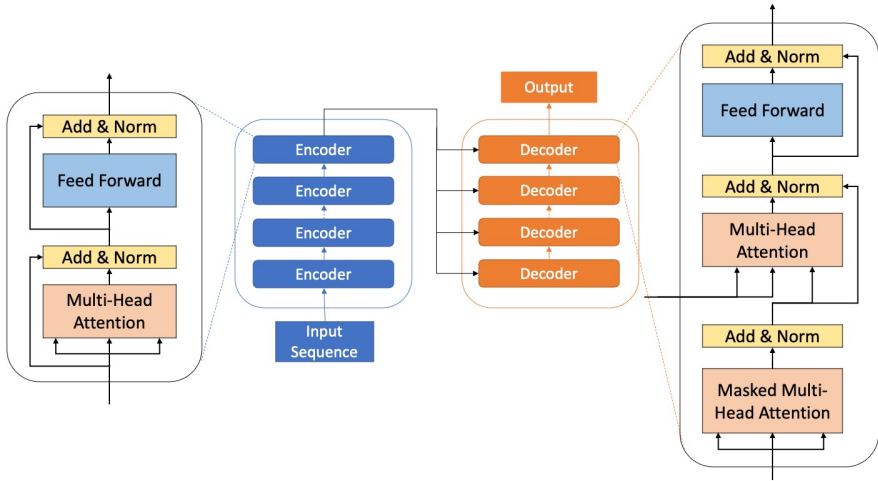
4 *Pre-Training Large Language Models*

Fig. 1 Transformer architecture. A Transformer consists of stacked blocks of encoder followed by a stacked blocks of decoders. The leftmost block is the detailed structure of an encoder block, and the rightmost block is the detailed structure of a decoder block.

GPT Architecture

The Generative Pre-trained Transformer (GPT) model is built on top of the decoder part of the Transformer model [19] that utilizes stacked pairs of a masked multi-head self-attention layer and a feed-forward layer (Figure 2, right). Unlike BERT, the GPT architecture has unidirectional attention where every token can only attend to its prior tokens. The GPT model leverages different types of input transformations for different tasks.

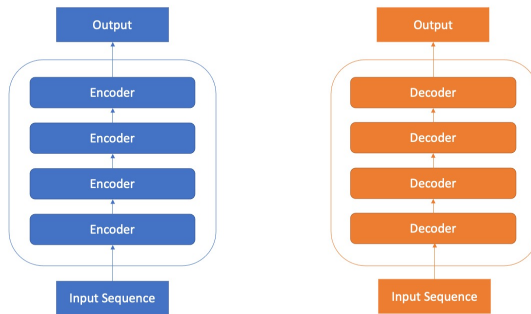


Fig. 2 BERT (left) and GPT (right) Architectures.

Scaling Laws

One key argument for the rise of deep learning (DL) is that the accuracy of DL models scales with the amount of data. This scaling property was not fully demonstrated until the emergence of Transformer-based LLMs. It has been shown [20] that the loss of LLMs scales with both the training data and model parameters; consequently, it scales with the amount of computation.

Specifically, the total number of floating point operations (FLOPs) is approximately,

$$T_{\text{FLOPs}} \sim 6 * P * D \quad (1)$$

where P and D are number of model parameters and tokens, respectively. This can be intuitively derived from the Transformer architecture (see Fig. 1): compared to the attention and other blocks, the feed forward block typically requires the most computation [21]. For each element of a feed-forward weight matrix, there are a total of 6 FLOPs (1 multiply-add operation during forward pass and 2 multiply-add operations during the backward pass) per input token. Coupled with the optimal training guidance [16], the computation hence scales quadratically with the model size.

In terms of distributed training of LLMs, numerous techniques have been introduced to enhance training speed by leveraging various parallelism approaches [21–23], reducing memory consumption [24], and employing lower precision computations [25]. Notably, the Zero Redundancy Optimizer (ZeRO) [23] enhances vanilla data parallelism by reducing memory usage for gradients (stage 1), optimizer states (stage 2), or offloading (stage 3). Gpipe [22] introduces pipeline parallelism, while Megatron [21] proposes tensor parallelism. Most LLMs are pre-trained using these aforementioned techniques, which we will further explore within two popular software frameworks.

3 Methodology

In this section, we will detail our PyTorch-based distributed training methods (TensorFlow’s LLM training is more tailored for TPU-based platforms). These methods are PyTorch’s fully sharded data parallel (FSDP) [26] and EleutherAI’s DeepSpeed-Megatron implementation [27], which are broadly representative of the standard approaches in the field. Additionally, we provide our cost analysis for compute and energy projections.

We use the following notations:

- M — the overall memory usage per device.
- C — communication size per device per batch step.
- DP, PP, and TP — the number of parallelisms assigned for data, pipeline, and tensor, respectively.
- T and R — the FLOP counts and measured performance in FLOPs per second, respectively.
- t and E — the runtime and energy usage, respectively.

6 Pre-Training Large Language Models

- b — the required communication bandwidth.

3.1 Distributed training

There are several major parallelisms in distributed training of LLM: 1) data parallel among model replica, i.e., each model replica processes a different data batch in the forward pass and then synchronizes in the backward pass. 2) pipeline parallel among model layers, i.e., pipelining the execution of a data batch through different layers of a model replica. 3) tensor parallel within layers, i.e., parallel processing of a layer by tensor decomposition. The memory usage, M , consists of 4 sub-components:

$$M = M_p + M_o + M_g + M_e, \quad (2)$$

where M_p , M_o , and M_g are for model parameters, optimizer states, and gradients, respectively. The extra overhead M_e stems from intermediate storage, such as activation maps or collections of parameter shards. Specifically, the value of these terms depends on the distributed training methods, and we use M_A and M_B to represent the overall memory usage per device for FSDP and DeepSpeed-Megatron, respectively. The details of M_A and M_B will be discussed in the following section.

3.1.1 FSDP

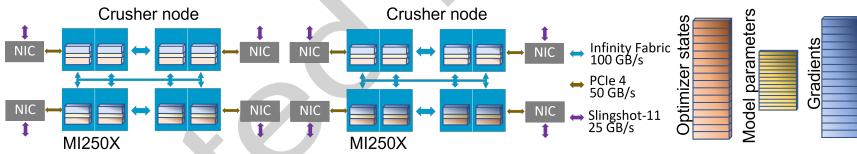


Fig. 3 Illustration of FSDP on 2 Crusher nodes. The optimizer states, model parameters, and gradients are evenly distributed among all 16 GPUs.

PyTorch’s fully sharded data parallel (FSDP) [26] is an enhancement over the standard distributed data parallel (DDP) [28], being optimized for more efficient memory usage. Instead of each device storing a full replica of the model, the model parameters, optimizer states, and gradients are fully sharded among all devices participating in data parallel training, as shown in Fig. 3 (based on Crusher architecture [29]. More details in Sec. 4). Effectively, FSDP is equivalent to ZeRO [23] for data parallel training, in which the sharding can be controlled at a finer-grained level. On the other hand, FSDP API is shipped within the PyTorch distribution itself and benefits from built-in support. Since PyTorch is supported on both the Nvidia CUDA and AMD ROCm stacks, FSDP functions smoothly on both the Summit [30] and Crusher [29] platforms as well.

With FSDP enabled, the memory usage per device now follows as,

$$\begin{aligned} M_A &= M_{\text{total}}/N_{\text{devices}} \\ &\sim 6 * M_p/N_{\text{devices}}, \end{aligned} \quad (3)$$

considering the size of gradients, optimizer states (e.g. Adam [31]), and overheads (FSDP units) to be about 1x, 2x, and 2x of the size of model parameters. This calculation assumes that all components have the same data type and that activation maps are re-computed via checkpointing [32]). However, the memory saving does come at a cost. Compared to DDP, where only an **Allreduce** during each backward propagation is required, FSDP requires an **Allgather** for the forward pass and both an **Allgather** and a **ReduceScatter** for the backward pass. Therefore, the per device communication volume per batch step, denoted by C_A , is $1.5\times$ of DDP's (**Allreduce** is implemented via a **ReduceScatter** and an **Allgather**),

$$C_A \sim 3 * M_p. \quad (4)$$

3.1.2 DeepSpeed-Megatron

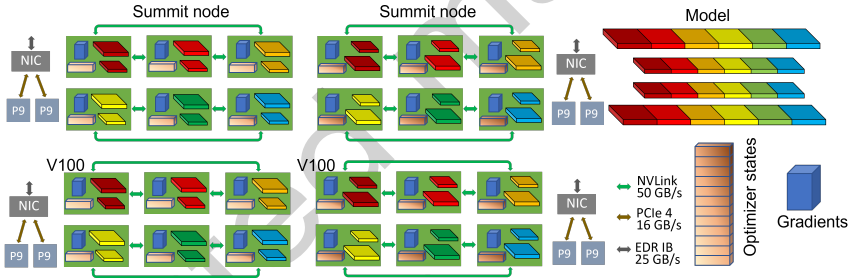


Fig. 4 Illustration of DeepSpeed-Megatron on 4 Summit nodes with tensor parallel at 6, pipeline parallel at 2, and data parallel (ZeRO stage 1) at 2.

A more matured stack for LLM training is DeepSpeed [33] and Megatron [21], where pipeline (PP) and tensor (TP) parallelisms are supported, in addition to sharded data (DP) parallelism. As shown in Fig. 4 (based on Summit architecture [30]. More details in Sec. 4), a $PP = 2, TP = 6, DP = 2$ parallel setting on 4 Summit nodes is illustrated. The tensor parallelism is within a node, where each layer is split. The pipeline parallelism is divided into 2 stages, each on a node with half of model layers. Finally, the ZeRO data parallelism is among the remaining level of parallelisms (i.e., $N_{\text{devices}}/(PP * TP)$), and typically only the optimizer states are sharded (ZeRO stage 1) to avoid excessive communication cost. Generally, the memory requirement per device follows,

$$M_B = M_p/(PP * TP) + M_o/DP$$

8 *Pre-Training Large Language Models*

$$\begin{aligned}
& + M_g / (PP * TP) + M_e \\
& \sim \frac{2 * (DP + PP * TP + 1)}{N_{\text{devices}}} M_p,
\end{aligned} \tag{5}$$

assuming the model layers are of similar sizes, the Adam optimizer is employed, and the overhead is of the same ratio as FSDP's. Because the scaling efficiencies of pipeline and tensor parallelism are worse than data parallelism (that is, they require more communication), the best scaling strategy is to find the minimum number of nodes that can fit the model in memory with model parallelism, and to apply the remaining resources to data parallelism. Based on the assumptions that 1) the communication volume of Zero-1 stage [23] is the same as data parallelism, 2) pipeline parallelism incurs orders of magnitude smaller [33] communication size compared to data parallelism, and 3) the tensor parallelism is within a sub-communication group of size T, the average per device communication volume for each batch step is,

$$C_B \sim \left(\frac{TP}{N_{\text{devices}}} + 1 \right) * 2 * M_p. \tag{6}$$

Compared to FSDP, DeepSpeed-Megatron has finer control of the parallelism and hence can achieve better scaling when properly tuned, especially at large scale. On the other hand, as third-party libraries for PyTorch, the support for different hardware and underlying compute libraries requires additional effort for porting and maintenance.

Port to AMD stack EleutherAI's GPT-NeoX [27] implementation of DeepSpeed-Megatron is an open community effort for LLM training. It doesn't yet officially support AMD GPUs and, to the best of our knowledge, we are the first effort to report its performance on the AMD DL stack.

3.2 Cost analysis

In this section, we outline our analysis method and metrics for runtime and energy projection, based on the scaling law (see Sec. 2) of training LLMs.

3.2.1 Computational requirements

As shown in Fig. 5, the total number of compute operations needed for the optimal training (20 tokens per parameter) [16] of LLMs is quadratically proportional to the number of model parameters (P). For example, training a 175B parameter GPT3 model requires 3.7×10^{24} FLOPs, and it quickly grows to 1.2×10^{26} FLOPs for a 1T parameter GPT-style model. This can serve as a quick heuristic to maximize the return of LLM modeling under a compute budget constraint. For a given model size (with P parameters), the region below the solid line (blue region in Fig. 5) indicates the data size is insufficient (i.e., the number of tokens is less than 20P). Conversely, for a given compute budget, the region above the solid line (yellow region, token-to-parameter ratio

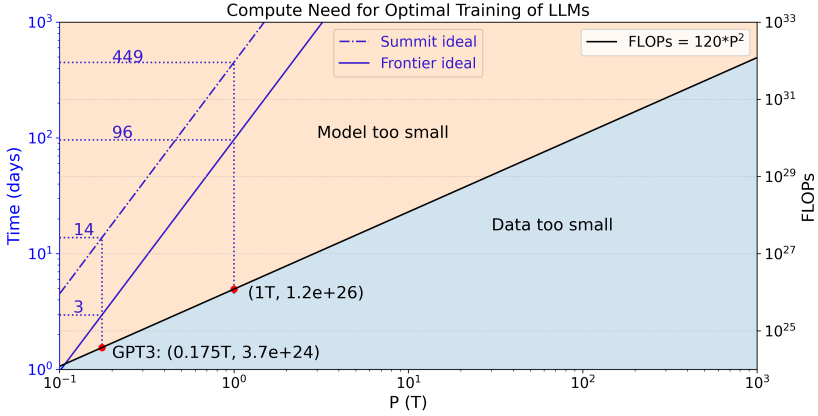


Fig. 5 Computation need (FLOPs) and ideal training time (days) assuming peak performance and perfect scaling for optimal training of LLMs on Summit and Frontier, respectively.

greater than 20) means the output accuracy would likely benefit from increasing the model size. Note that the recent study [9] suggests a significantly higher ratio, indicating that the computation requirements for optimal training will correspondingly increase.

3.2.2 Runtime and energy projection

Like any standard DL modeling, the training of LLMs is an iterative process, where data are processed in batch steps. Unlike the complications which arose in large-scale convolution neural networks – e.g., the convergence issue of large-batch training of ResNet50 on ImageNet [34] – LLMs training is very stable owing to both the Transformer-based model architecture and the large data size (tokens). Therefore, the runtime can be straightforwardly predicted via,

$$\begin{aligned} t &= T_{\text{FLOPs}} / R_{\text{FLOPs}} \\ &\sim 120 * P^2 / R_{\text{FLOPs}}, \end{aligned} \quad (7)$$

where T_{FLOPs} and R_{FLOPs} are the compute operations (See Fig. 5) in FLOPs and training performance in FLOPs (measured from about 100 iterations excluding the warm-up). The energy consumption can be evaluated by,

$$E = t * R_{\text{Watt}}, \quad (8)$$

where R_{Watt} is the averaged power measured from few iterations.

The ideal training times on Summit and Frontier are also plotted in Fig. 5, given the peak performance in half precision for the V100 (112 TFLOPs) and MI250X (384 TFLOPs) GPUs and linear scaling up to full system. In practice, the optimal criteria may not be able to be satisfied, especially for

Table 1 The specification of GPT-style LLMs in the evaluation.

Model	Number of layers	Hidden size	Number of attention heads	Sequence length	Vocab size
1.4B	24	2,064	24	2,048	~50,000
16B	48	5,184	48		
175B	96	12,288	96		
1T	128	25,600	128		

models of trillions of parameters. In the case of comparable size of data for a 1T parameter model, the ideal training time on Frontier would be one week.

4 Evaluation

In this section, we will detail our experimental setup, hardware and software stack, performance tuning, and scaling results. We also provide a trace-based analysis of power usage, and projections of runtime and energy cost. Our experiments are performed on two preeminent but architecturally different leadership computing platforms, Summit and Frontier, which have Nvidia and AMD GPUs, respectively. Each are representative in their own right of today’s HPC landscape.

4.1 Experiment setup

Hardware To evaluate the cost of pre-training LLMs, we execute two code bases (FSDP and DeepSpeed-Megatron) on both the Summit supercomputer [30] and on the Crusher [29] early access testbed system for Frontier. Each Summit node (see Fig. 4) is equipped with six NVIDIA V100 GPUs and two IBM Power9 CPUs. Groups of three V100s are fully connected with NVLink fabric with 50 GB/s of bandwidth, and the nodes themselves use an EDR InfiniBand interconnect with 25 GB/s of bandwidth. Crusher is a development system with an architecture identical to Frontier’s but at a smaller scale (192 vs 9408 nodes). Each Crusher node (see Fig. 3) is equipped with four AMD Instinct MI250X GPUs with dual Graphics Compute Dies (GCD) and one third-generation EPYC CPU. A GCD is viewed as an effective GPU, and we use GCD and GPU interchangeably in the following discussion. All four MI250X GPUs (eight effective GPUs) are connected using 50 GB/s Infinity Fabric, and the nodes are connected via a Slingshot-11 interconnect with 100 GB/s of bandwidth.

Software We use PyTorch v1.12.1 to evaluate both FSDP and GPT-NeoX [27]. CUDA v11.0.3, cuDNN v8.0.4, and NCCL v2.8.4 are utilized on Summit, and ROCm v5.1.0, MIOpen v2.14.0, and RCCL v2.11.4 with libfabric v1.15.0 plugin are utilized on Crusher.

In the following, we will systematically evaluate the performance, scaling, runtime, and energy efficiency of these two training frameworks on two leadership-class supercomputers, and draw considerations for future platforms.

4.2 Results

Because the runtime projection (Eq. 7) is based on per-device computation performance, we first investigate the performance optimization for the training frameworks on both Nvidia- and AMD- based platforms. The primary tuning knobs include the allocation of data, pipeline, and tensor parallelism, operation fusion for DeepSpeed-Megatron, and parameter prefetching for FSDP. All computations are performed in half precision (FP16).

4.2.1 Performance tuning

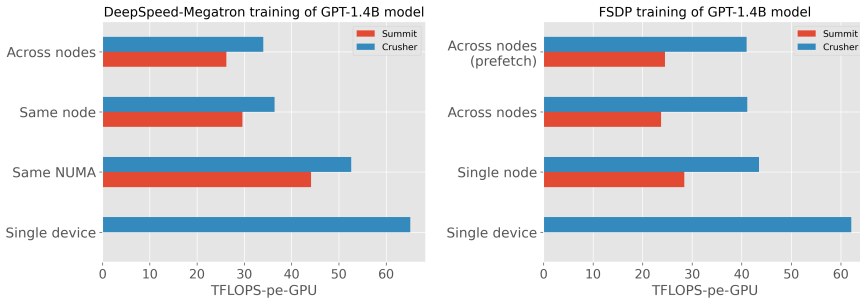


Fig. 6 Performance (TFLOPS) of training GPT-1.4B on V100-based Summit and MI250X-based Crusher. (left) DeepSpeed-Megatron: Single device (1 GCD of MI250X); Same NUMA (DP=PP=1, TP=3 for Summit, 2 for Crusher); Same node (DP=PP=1, TP=6 for Summit, 8 for Crusher); Across nodes (DP=2, PP=1, TP=6 for Summit, 8 for Crusher). (right) FSDP. Single device (1 GCD of MI250X); Single node (1 Summit/Crusher node); Across nodes (2 Summit/Crusher nodes); Across nodes (with parameter prefetching).

High-performance computing platforms typically consist of various affinity domains of different interconnect capabilities (See Fig. 3, 4). For example, within a non-uniform memory access (NUMA), 3 V100s are connected with NVLink of 50 GB/s on Summit, and 2 GCDs of MI250X are connected with Infinity Fabric of 200 GB/s on Crusher, while for across nodes the numbers drops to 25 GB/s and 100 GB/s, respectively. The performance of training LLMs are extremely sensitive to communication, and aligning model parallelism to different affinities has a significant impact on performance. As shown in Fig. 6, for a single GPU (1 GCD of MI250X), the training performance for FSDP and DeepSpeed-Megatron is 62.1 and 65.1 TFLOPS, respectively. From this baseline, the performance drops 20% and 44% for DeepSpeed-Megatron when using tensor parallelism within a NUMA domain and a node on Summit, respectively. The impact for FSDP is less (30% within a node compared to 44% for DeepSpeed-Megatron) due to less frequent communication and a smaller total message size (Eq. 4), although the benefit of parameter prefetching is negligible for the model size. One thing to note is that the model architecture affects the computation performance as well because different matrix sizes

result in different performance for general matrix multiplication (GEMM) kernels [35], which are the computational workhorse of Transformers. For example, if we fix the number of layers for GPT-1.4B model and adjust the hidden size, we observe that performance improves to 77 TFLOPS per MI250X GCD for the hidden size 2304 and a total of 1.76B parameters. We consider this as the achievable performance for training LLMs on the platform.

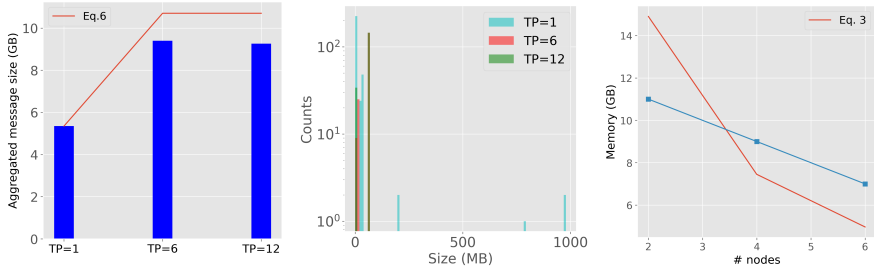


Fig. 7 The per-batch-step aggregated message size (left) and histogram (middle) of DeepSpeed-Megatron training of GPT-1.4B model with DP=PP=1, and TP=1, 6, and 12 on Summit. (right) the peak memory usage of FSDP training of GPT-16B model on Summit. The red lines are Eq. 6 and Eq. 3 for message size and memory, respectively.

To better understand the communication traffic and memory usage, we plot the per-step aggregated message size and histogram for the DeepSpeed-Megatron training of GPT-1.4B model, and the memory usage for FSDP training of GPT-16B model. Compared to our projections (Eq. 6 and Eq. 3), the measurements are consistent. The message volume is about $1.8\times$ the size of standard data parallelism (DP=PP=TP=1) and the communication pattern changed from few large messages (198, 783, 975 MB) to more frequent medium messages (64 MB). Likewise, the per-device memory for FSDP is roughly the total memory usage evenly divided among all devices.

4.2.2 Scaling Analysis

Following on the previous observations, we scale up the LLMs training on both Summit and Crusher, as shown in Fig. 8. The minimum model-parallel scale required to fit the GPT-16B model on Summit is 2 nodes (based on Eq. 3, about 180GB is needed). Consequently, the model parallelism for DeepSpeed-Megatron training (PP,TP, see Sec. 3 for details) can be assigned as (1,12) or (2,6), considering TP is more efficient within a node than PP. The measured performance is 12.9 versus 10.9 TFLOPS at 8 nodes for (1,12) and (2,6), respectively. Therefore, additional scaling, we use (1,12) for model parallelism, and the remaining resources are assigned to data parallelism. To further improve the scaling, we increased batch size by 2 (because the ZeRO-1 DP further reduce the memory footprint when more devices are used) starting from 16 nodes to 256 nodes. For this model size, the best per-device performance

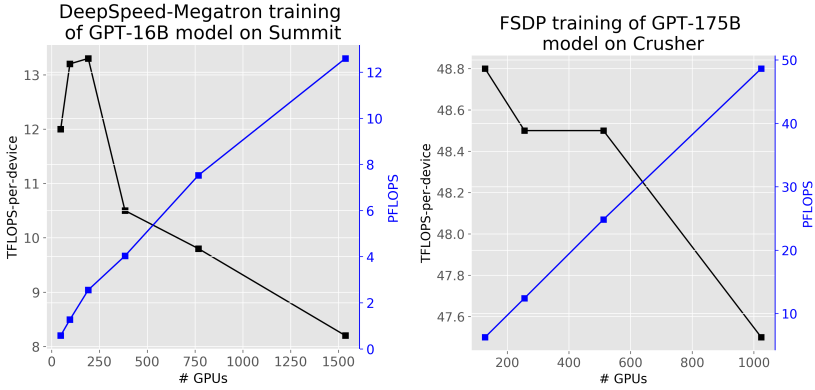


Fig. 8 The scaling of (left) DeepSpeed-Megatron training of GPT-16B model on Summit up to 1536 V100s; (right) FSDP training of GPT-175B on Crusher up to 512 MI250Xs (1024 effective GPUs).

is reached at 32 nodes (192 V100s), and the aggregate 13 PFLOPS training performance is obtained with 1536 V100s.

Similarly, we evaluate FSDP training of GPT-175B model on Crusher (see Fig. 8). Compared to GPT-1.4B model (see Fig. 6), the per-device performance actually increases by 17%, indicating there exists a tipping point between the trade-off of more computation and more communication. Up to 128 nodes (1024 effective GPUs), the scaling efficiency is about 97%, signaling Frontier can be a promising platform for training LLMs of these model sizes.

4.2.3 Power, Memory, and Utilization Trace

To estimate the energy usage, we trace the GPU power in watts during the training. In Fig. 9, a timeline trace of GPU power, memory, and utilization are plotted for FSDP training of GPT 175B model on Summit and Crusher, respectively. Note that on Crusher the memory and utilization is for 1 GCD while the power trace is for 1 MI250X (2 GCDs). One batch step takes 359 and 301 seconds, correspondingly, being depicted in the figure by the interval between the dashed lines. All quantities show clear periodical behavior. The memory usage oscillates because of the frequent allocation and deallocation of memory to fit large models. The utilization trace is different for AMD GPU due to the frequency of sampling. Because the NCCL/RCCCL communication utilizes GPU as well, the power is more representative of the load of computation. The averaged power usage is about 85 and 408 Watts, for Summit and Crusher, respectively, and the corresponding computational efficiency is 0.165 and 0.235 TFLOPS/Watt.

Similarly, we analyze the power, memory, and utilization trace for training a GPT-1T parameter model with DeepSpeed-Megatron on Summit and FSDP on Crusher, respectively. As shown in Fig. 10, one batch step takes 341 and 1290 seconds (according to the log), respectively, on the two platforms and the two frameworks behave similarly. Compared to training the GPT 175B

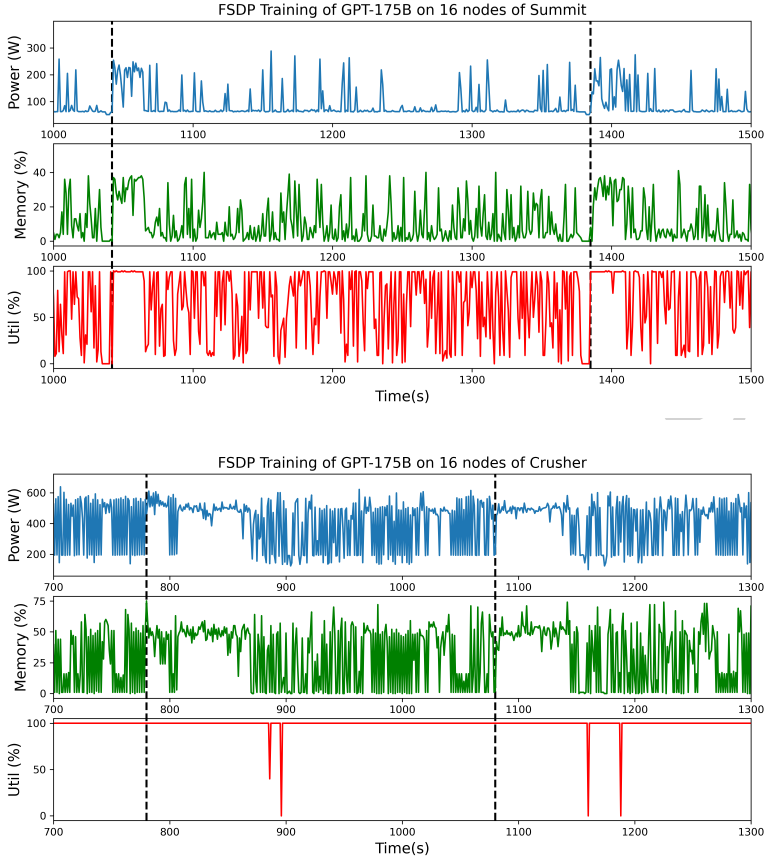


Fig. 9 The GPU power, memory, and utilisation trace of FSDP training of a GPT 175B model on 16 Summit nodes and 16 Crusher nodes, respectively. The time interval between the dashed lines indicates one batch step.

parameter model, the power usage is significantly lower (on average, 66 for Summit and 151 Watts for Crusher), indicating lower computation loads, while the utilization stays at 100% because of the large amount of communication in terms of both message size and frequency. Nevertheless, due to the benefit of the large per-device memory, Crusher is 2 orders of magnitude more efficient for training GPT 1T model.

4.2.4 Runtime and energy projection

Following Eq. 7 and 8, we project the training time and energy consumption based on the previously discussed performance and scaling analyses. In Table. 2, the estimations are listed for GPT-style model of sizes from 1.4B to 1T on both Summit and Frontier (using linear extrapolation based on Crusher for the latter system). For 1.4B and 16B models, the total computation needs are

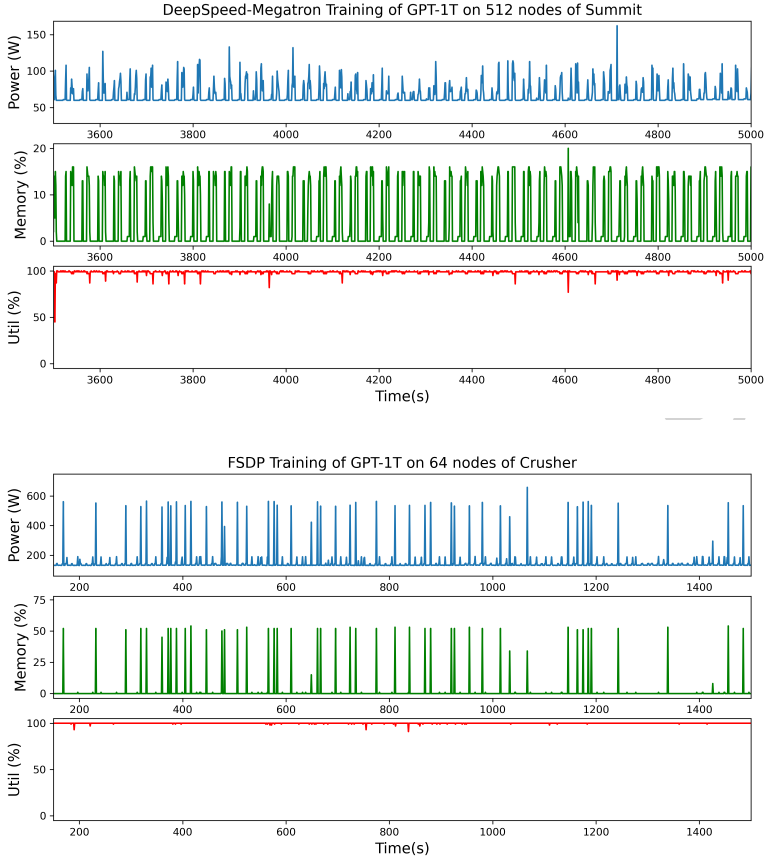


Fig. 10 The GPU power, memory, and utilization trace of training of GPT 1T model: (upper) with DeepSpeed-Megatron on 512 Summit nodes; (lower) with FSDP on 64 Crusher nodes.

based on the optimal training recommendation [16], where data size (tokens) is $20\times$ the number of model parameters. For larger models (175B and 1T), considering the limitation of data, a range is provided with the lower and upper bound being $1\times$ and $20\times$ of the model parameters. Given the per-device computation performance (TFLOPS) and the averaged power (watts), the power efficiency is also estimated. In general, MI250X-based Crusher is more energy-efficient for training LLMs, and the advantage grows rapidly for model sizes beyond GPT3.

As Crusher system bears the same architecture as the first Exascale system—Frontier and their unprecedented mix-precision capability, we believe they are well-suited as the platform for training LLMs at extreme scale. We observe that these values for training LLMs are over $4\times$ more power efficient than executing traditional simulation workloads (~ 52 GFLOPS/Watt).

Table 2 Training time and energy cost projection for GPT-style LLMs on Summit and Frontier supercomputers.

Model	Summit			Frontier		
	Time (node hours)	Energy (MWh)	Efficiency (TFLOPS/Watt)	Time (node hours)	Energy (MWh)	Efficiency (TFLOPS/Watt)
1.4B	202	2.8	0.204	108	2.2	0.255
16B	23K	30	0.185	12K	24	0.239
175B	[0.6M, 12M]	[309, 6176]	0.165	[0.1M, 3M]	[217, 4338]	0.235
1T	[1B, 21B]	[3.7e5, 7e6]	0.01	[8M, 160M]	[5e3, 9e4]	0.343

Compared to the cost modeling based on the theoretical performance in Fig. 5), our projection is about one order of magnitude larger for GPT3-size model, which is due to software maturity and the hardware limitation. We will discuss the performance gap, current practices and provide considerations for future platforms.

One caveat to consider in our estimation is that the analysis is based on the current implementations of GPT-NeoX (DeepSpeed-Megatron) and PyTorch (FSDP). It's important to note that the field is rapidly evolving, with ongoing advancements that can further reduce communication costs. For instance, techniques such as message compression [36], model pruning [37], and autotuning of parallel strategies [38] have shown promise in significantly mitigating the network bandwidth requirements for scaling up LLM training. If these techniques are validated in terms of model accuracy and generalizability, it will be necessary to update the computational and energy cost projections accordingly. In our future study, we will delve into these optimizations and conduct end-to-end pre-training of LLMs.

4.3 Considerations for future platforms

Based on the above evaluation, we can draw following observations on the considerations for future platforms.

Compared to the theoretical peak performance of a device (V100 or MI250X), there is a gap in how much performance that can be extracted for training LLMs, especially at scale. In Fig. 11, we plotted TFLOPs per device and projected training time for GPT 1T models with 1T tokens on both Summit and Frontier, respectively. The achievable performance is around 40% of the peak, similar for V100 and MI250X. This reflects the kernel workloads (e.g., GEMM parameters) of the model architecture. However, accommodating a large mode comes at the cost of additional communication (Fig. 4, 6), and the performance at scale has another $3\times$ and 2 orders of magnitude drop for Frontier and Summit, respectively. The impact is larger on Summit because 1) the per node GPU memory on Summit is $5\times$ smaller and hence more nodes are needed to fit in the model; 2) the node injection bandwidth is $4\times$ smaller. Therefore, for LLM training, larger amounts of per node device memory and communication bandwidth lead to obtaining better performance, but the specific requirements depend on the model size. In the following, we will

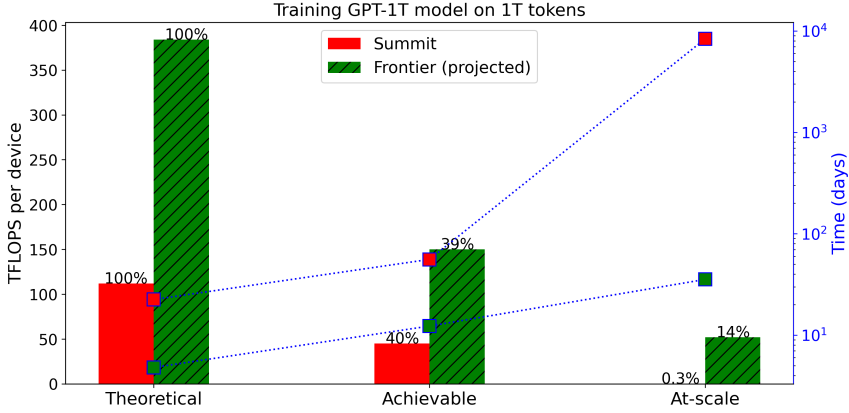


Fig. 11 Performance gap of training GPT 1T model at scale.

extrapolate the communication requirements for training a GPT 1T model on MI250X-based platforms with 128 GB of memory per dual-die GPU.

For a GPT 1T model using a sequence length of 2048 tokens, one MI250X can fit 4 samples per batch. Hence the ideal step time with linear scaling is,

$$\begin{aligned}
 t^{\text{step}} &= T_{\text{FLOPS}}^{\text{step}} / R_{\text{FLOPS}} \\
 &= 4.9 * 10^4 / R_{\text{TFLOPS}}.
 \end{aligned} \tag{9}$$

Assuming the software is perfectly optimized to overlap the communication with computation, the minimum bandwidth required to achieve perfect scaling following the communication volume C_A (Eq. 4) is then,

$$\begin{aligned}
 b_{\text{ideal}}(\text{GB/s}) &= C_A / t^{\text{step}} \\
 &= 0.24 * R_{\text{TFLOPS}}.
 \end{aligned} \tag{10}$$

Similarly, a bandwidth requirement for the scenario of overlapping half of communication with computation can also be derived. As shown in Fig. 12, the lower-bound requirement for ideal scaling of distributed training of GPT 1T model on MI250X-based platforms is plotted. For theoretical peak and achievable performance, the minimum per-device communication bandwidth needed is 37 and 94 GB/s, respectively. The current 25 GB/s per-device on Crusher is not sufficient to support linear scaling for training GPT 1T model. We ignore I/O requirement in our analysis because it can be straightforwardly hidden among computations given the typical global batch size of millions of tokens (less than GB in size) and an ideal step time of hundreds of seconds.

The above analysis can be applied to future generation GPU devices with specified memory capacity and compute performance as well, and hopefully can serve as a heuristic for platform design.

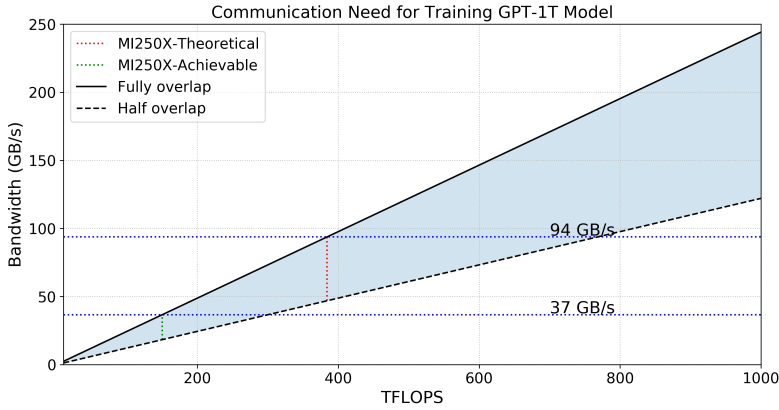


Fig. 12 Communication need for distributed training of GPT 1T model on MI250X-based platforms.

5 Conclusion

To conclude, we systematically evaluated state-of-the-art techniques for training LLMs on two of the DOE's leadership-class platforms. By establishing performance baselines and scaling up the model sizes and the computing resources, we projected the runtime and energy cost for training LLMs from scratch. This can serve as a base for the feasibility analysis and practical guide in building foundation models for sciences. Our results indicate that Frontier is a promising platform for GPT3-size model training. Furthermore, the considerations we drew from the evaluations of both Nvidia- and AMD-based hardware and software stacks can benefit the design of future platforms.

Acknowledgments. J.Y. would like to thank Quentin Anthony and Stella Biderman from EleutherAI, and Less Wright and Geeta Chauhan from PyTorch team for the valuable discussions. This research was sponsored by and used resources of the Oak Ridge Leadership Computing Facility (OLCF), which is a DOE Office of Science User Facility at the Oak Ridge National Laboratory supported by the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Declarations

Ethical Approval

Not applicable.

Competing interests

The authors declare no competing interests.

Authors' contributions

J.Y. performed the model training and analysis. J.Y. and S.D. wrote the main manuscript text, and J.G. and F.W. edited the manuscript. The project was advised by F.W. and G.T.. All authors discussed the results and reviewed the manuscript.

Funding

This research was sponsored by and used resources of the Oak Ridge Leadership Computing Facility (OLCF), which is a DOE Office of Science User Facility at the Oak Ridge National Laboratory supported by the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

Availability of data and materials

The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

References

- [1] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. *Advances in neural information processing systems* **30** (2017)
- [2] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018)
- [3] Radford, A., Narasimhan, K.: Improving language understanding by generative pre-training. (2018)
- [4] Bommasani, R., Hudson, D.A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M.S., Bohg, J., Bosselut, A., Brunskill, E., Brynjolfsson, E., Buch, S., Card, D., Castellon, R., Chatterji, N.S., Chen, A.S., Creel, K., Davis, J.Q., Demszky, D., Donahue, C., Doumbouya, M., Durmus, E., Ermon, S., Etchemendy, J., Ethayarajh, K., Fei-Fei, L., Finn, C., Gale, T., Gillespie, L., Goel, K., Goodman, N.D., Grossman, S., Guha, N., Hashimoto, T., Henderson, P., Hewitt, J., Ho, D.E., Hong, J., Hsu, K., Huang, J., Icard, T., Jain, S., Jurafsky, D., Kalluri, P., Karamcheti, S., Keeling, G., Khani, F., Khattab, O., Koh, P.W., Krass, M.S., Krishna, R., Kuditipudi, R., et al.: On the opportunities and risks of foundation models. *CoRR* **abs/2108.07258** (2021) <https://arxiv.org/abs/2108.07258>
- [5] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D.,

- Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., Amodei, D.: Language models are few-shot learners. In: Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M.F., Lin, H. (eds.) *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901. Curran Associates, Inc., virtual (2020). <https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf>
- [6] OpenAI: GPT-4 Technical Report (2023)
- [7] Ramesh, A., Pavlov, M., Goh, G., Gray, S., Voss, C., Radford, A., Chen, M., Sutskever, I.: Zero-Shot Text-to-Image Generation. *arXiv* (2021). <https://doi.org/10.48550/ARXIV.2102.12092>. <https://arxiv.org/abs/2102.12092>
- [8] Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X.V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P.S., Sridhar, A., Wang, T., Zettlemoyer, L.: OPT: Open Pre-trained Transformer Language Models. *arXiv* (2022). <https://doi.org/10.48550/ARXIV.2205.01068>. <https://arxiv.org/abs/2205.01068>
- [9] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., Lample, G.: LLaMA: Open and Efficient Foundation Language Models (2023)
- [10] Scao, T.L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A.S., Yvon, F., Gallé, M., Tow, J., Rush, A.M., Biderman, S., Webson, A., Ammanamanchi, P.S., Wang, T., Sagot, B., Muennighoff, N., del Moral, A.V., Ruwase, O., Bawden, R., Bekman, S., McMillan-Major, A., Beltagy, I., Nguyen, H., Saulnier, L., Tan, S., Suarez, P.O., Sanh, V., Laurençon, H., Jernite, Y., Launay, J., Mitchell, M., Raffel, C., Gokaslan, A., Simhi, A., Soroa, A., Aji, A.F., Alfassy, A., Rogers, A., Nitzav, A.K., Xu, C., Mou, C., Emezue, C., Klammer, C., Leong, C., van Strien, D., Adelani, D.I., Radev, D., Ponferrada, E.G., Levkovizh, E., Kim, E., Natan, E.B., De Toni, F., Dupont, G., Kruszewski, G., Pistilli, G., Elsahar, H., Benyamina, H., Tran, H., Yu, I., Abdulmumin, I., Johnson, I., Gonzalez-Dios, I., de la Rosa, J., Chim, J., Dodge, J., Zhu, J., Chang, J., Frohberg, J., Tobing, J., Bhattacharjee, J., Almubarak, K., Chen, K., Lo, K., Von Werra, L., Weber, L., Phan, L., allal, L.B., Tanguy, L., Dey, M., Muñoz, M.R., Masoud, M., Grandury, M., Šaško, M., Huang, M., Coavoux, M., Singh, M., Jiang, M.T.-J., Vu, M.C., Jauhar, M.A., Ghaleb, M., Subramani, N., Kassner, N., Khamis, N., Nguyen, O., Espejel, O., de Gibert, O., Villegas, P., Henderson, P., Colombo, P., Amuok, P., Lhoest, Q., Harlman, R., Bommasani, R., López, R.L., Ribeiro, R., Osei, S., Pyysalo, S., Nagel, S., Bose, S., Muhammad, S.H., Sharma, S.,

Longpre, S., Nikpoor, S., Silberberg, S., Pai, S., Zink, S., Torrent, T.T., Schick, T., Thrush, T., Danchev, V., Nikoulina, V., Laippala, V., Lepercq, V., Prabhu, V., Alyafeai, Z., Talat, Z., Raja, A., Heinzerling, B., Si, C., Salesky, E., Mielke, S.J., Lee, W.Y., Sharma, A., Santilli, A., Chaffin, A., Stiegler, A., Datta, D., Szczechla, E., Chhablani, G., Wang, H., Pandey, H., Strobelt, H., Fries, J.A., Rozen, J., Gao, L., Sutawika, L., Bari, M.S., Al-shaibani, M.S., Manica, M., Nayak, N., Teehan, R., Albanie, S., Shen, S., Ben-David, S., Bach, S.H., Kim, T., Bers, T., Fevry, T., Neeraj, T., Thakker, U., Raunak, V., Tang, X., Yong, Z.-X., Sun, Z., Brody, S., Uri, Y., Tojarieh, H., Roberts, A., Chung, H.W., Tae, J., Phang, J., Press, O., Li, C., Narayanan, D., Bourfoune, H., Casper, J., Rasley, J., Ryabinin, M., Mishra, M., Zhang, M., Shoeybi, M., Peyrounette, M., Patry, N., Tazi, N., Sanseviero, O., von Platen, P., Cornette, P., Lavallée, P.F., Lacroix, R., Rajbhandari, S., Gandhi, S., Smith, S., Requena, S., Patil, S., Dettmers, T., Barua, A., Singh, A., Cheveleva, A., Ligozat, A.-L., Subramonian, A., Névoul, A., Lovering, C., Garrette, D., Tunuguntla, D., Reiter, E., Taktasheva, E., Voloshina, E., Bogdanov, E., Winata, G.I., Schoelkopf, H., Kalo, J.-C., Novikova, J., Forde, J.Z., Clive, J., Kasai, J., Kawamura, K., Hazan, L., Carpuat, M., Clinciu, M., Kim, N., Cheng, N., Serikov, O., Antverg, O., van der Wal, O., Zhang, R., Zhang, R., Gehrmann, S., Pais, S., Shavrina, T., Scialom, T., Yun, T., Limisiewicz, T., Rieser, V., Protasov, V., Mikhailov, V., Pruksachatkun, Y., Belinkov, Y., Bamberger, Z., Kasner, Z., Rueda, A., Pestana, A., Feizpour, A., Khan, A., Faranak, A., Santos, A., Hevia, A., Unldreaj, A., Aghagol, A., Abdollahi, A., Tammour, A., HajiHosseini, A., Behrooz, B., Ajibade, B., Saxena, B., Ferrandis, C.M., Contractor, D., Lansky, D., David, D., Kiela, D., Nguyen, D.A., Tan, E., Baylor, E., Ozoani, E., Mirza, F., Ononiwu, F., Rezanejad, H., Jones, H., Bhattacharya, I., Solaiman, I., Sedenko, I., Nejadgholi, I., Passmore, J., Seltzer, J., Sanz, J.B., Fort, K., Dutra, L., Samagaio, M., Elbadri, M., Mieskes, M., Gerchick, M., Akinlolu, M., McKenna, M., Qiu, M., Ghauri, M., Burynok, M., Abrar, N., Rajani, N., Elkott, N., Fahmy, N., Samuel, O., An, R., Kromann, R., Hao, R., Alizadeh, S., Shubber, S., Wang, S., Roy, S., Viguier, S., Le, T., Oyebede, T., Le, T., Yang, Y., Nguyen, Z., Kashyap, A.R., Palasciano, A., Callahan, A., Shukla, A., Miranda-Escalada, A., Singh, A., Beilharz, B., Wang, B., Brito, C., Zhou, C., Jain, C., Xu, C., Fourrier, C., Periñán, D.L., Molano, D., Yu, D., Manjavacas, E., Barth, F., Fuhrmann, F., Altay, G., Bayrak, G., Burns, G., Vrabec, H.U., Bello, I., Dash, I., Kang, J., Giorgi, J., Golde, J., Posada, J.D., Sivaraman, K.R., Bulchandani, L., Liu, L., Shinzato, L., de Bykhovetz, M.H., Takeuchi, M., Pàmies, M., Castillo, M.A., Nezhurina, M., Sängler, M., Samwald, M., Cullan, M., Weinberg, M., De Wolf, M., Mihaljcic, M., Liu, M., Freidank, M., Kang, M., Seelam, N., Dahlberg, N., Broad, N.M., Muellner, N., Fung, P., Haller, P., Chandrasekhar, R., Eisenberg, R., Martin, R., Canalli, R., Su, R., Su, R., Cahyawijaya, S., Garda, S., Deshmukh, S.S., Mishra, S., Kiblawi, S.,

- Ott, S., Sang-aaroonsiri, S., Kumar, S., Schweter, S., Bharati, S., Laud, T., Gigant, T., Kainuma, T., Kusa, W., Labrak, Y., Bajaj, Y.S., Venkatraman, Y., Xu, Y., Xu, Y., Xu, Y., Tan, Z., Xie, Z., Ye, Z., Bras, M., Belkada, Y., Wolf, T.: BLOOM: A 176B-Parameter Open-Access Multilingual Language Model. arXiv (2022). <https://doi.org/10.48550/ARXIV.2211.05100>. <https://arxiv.org/abs/2211.05100>
- [11] Beltagy, I., Cohan, A., Lo, K.: Scibert: Pretrained contextualized embeddings for scientific text. CoRR **abs/1903.10676** (2019) <https://arxiv.org/abs/1903.10676>
- [12] Trewartha, A., Walker, N., Huo, H., Lee, S., Cruse, K., Dagdelen, J., Dunn, A., Persson, K.A., Ceder, G., Jain, A.: Quantifying the advantage of domain-specific pre-training on named entity recognition tasks in materials science. Patterns **3**(4), 100488 (2022). <https://doi.org/10.1016/j.patter.2022.100488>
- [13] Hong, Z., Ajith, A., Pauloski, G., Duede, E., Malamud, C., Magoulas, R., Chard, K., Foster, I.: ScholarBERT: Bigger is Not Always Better. arXiv (2022). <https://doi.org/10.48550/ARXIV.2205.11342>. <https://arxiv.org/abs/2205.11342>
- [14] Horawalavithana, S., Ayton, E., Sharma, S., Howland, S., Subramanian, M., Vasquez, S., Cosbey, R., Glenski, M., Volkova, S.: Foundation models of scientific knowledge for chemistry: Opportunities, challenges and lessons learned. In: Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models, pp. 160–172. Association for Computational Linguistics, virtual+Dublin (2022). <https://doi.org/10.18653/v1/2022.bigscience-1.12>. <https://aclanthology.org/2022.bigscience-1.12>
- [15] Knoth, P., Zdrahal, Z.: Core: three access levels to underpin open access. D-Lib Magazine **18**(11/12) (2012)
- [16] Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D.d.L., Hendricks, L.A., Welbl, J., Clark, A., Hennigan, T., Noland, E., Millican, K., Driessche, G.v.d., Damoc, B., Guy, A., Osindero, S., Simonyan, K., Elsen, E., Rae, J.W., Vinyals, O., Sifre, L.: Training Compute-Optimal Large Language Models. arXiv (2022). <https://doi.org/10.48550/ARXIV.2203.15556>. <https://arxiv.org/abs/2203.15556>
- [17] Training a 1 Trillion Parameter Model With PyTorch Fully Sharded Data Parallel on AWS. <https://medium.com/pytorch/training-a-1-trillion-parameter-model-with-pytorch-fully-sharded-data-parallel-on-aws>
Accessed: 2022-11-11

- [18] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (2021). <https://openreview.net/forum?id=YicbFdNTTy>
- [19] Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al.: Improving language understanding by generative pre-training (2018)
- [20] Kaplan, J., McCandlish, S., Henighan, T., Brown, T.B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., Amodei, D.: Scaling laws for neural language models. CoRR **abs/2001.08361** (2020) <https://arxiv.org/abs/2001.08361>
- [21] Narayanan, D., Shoeybi, M., Casper, J., LeGresley, P., Patwary, M., Korthikanti, V., Vainbrand, D., Kashinkunti, P., Bernauer, J., Catanzaro, B., Phanishayee, A., Zaharia, M.: Efficient large-scale language model training on gpu clusters using megatron-lm. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC '21. Association for Computing Machinery, New York, NY, USA (2021). <https://doi.org/10.1145/3458817.3476209>. <https://doi.org/10.1145/3458817.3476209>
- [22] Huang, Y., Cheng, Y., Chen, D., Lee, H., Ngiam, J., Le, Q.V., Chen, Z.: Gpipe: Efficient training of giant neural networks using pipeline parallelism. CoRR **abs/1811.06965** (2018) <https://arxiv.org/abs/1811.06965>
- [23] Rajbhandari, S., Rasley, J., Ruwase, O., He, Y.: Zero: Memory optimizations toward training trillion parameter models. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC '20. IEEE Press, Atlanta, Georgia (2020)
- [24] Chen, T., Xu, B., Zhang, C., Guestrin, C.: Training deep nets with sublinear memory cost. CoRR **abs/1604.06174** (2016) <https://arxiv.org/abs/1604.06174>
- [25] Micikevicius, P., Narang, S., Alben, J., Diamos, G.F., Elsen, E., García, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., Wu, H.: Mixed precision training. CoRR **abs/1710.03740** (2017) <https://arxiv.org/abs/1710.03740>
- [26] Introducing PyTorch Fully Sharded Data Parallel (FSDP) API. <https://pytorch.org/blog/introducing-pytorch-fully-sharded-data-parallel-api/>. Accessed: 2022-11-19

- [27] Andonian, A., Anthony, Q., Biderman, S., Black, S., Gali, P., Gao, L., Hallahan, E., Levy-Kramer, J., Leahy, C., Nestler, L., Parker, K., Pieler, M., Purohit, S., Songz, T., Phil, W., Weinbach, S.: GPT-NeoX: Large Scale Autoregressive Language Modeling in PyTorch. <https://doi.org/10.5281/zenodo.5879544>. <https://www.github.com/eleutherai/gpt-neox>
- [28] Li, S., Zhao, Y., Varma, R., Salpekar, O., Noordhuis, P., Li, T., Paszke, A., Smith, J., Vaughan, B., Damania, P., Chintala, S.: Pytorch distributed: Experiences on accelerating data parallel training. *Proc. VLDB Endow.* **13**(12), 3005–3018 (2020). <https://doi.org/10.14778/3415478.3415530>
- [29] OLCF: Crusher. https://docs.olcf.ornl.gov/systems/crusher_quick_start_guide.html. Accessed: 2022-11-20 (2022)
- [30] OLCF: Summit. https://docs.olcf.ornl.gov/systems/summit_user_guide.html. Accessed: 2022-11-20 (2022)
- [31] Kingma, D., Ba, J.: Adam: A method for stochastic optimization. *International Conference on Learning Representations* (2014)
- [32] Chen, T., Xu, B., Zhang, C., Guestrin, C.: Training deep nets with sublinear memory cost. *CoRR* **abs/1604.06174** (2016) <https://arxiv.org/abs/1604.06174>
- [33] Rasley, J., Rajbhandari, S., Ruwase, O., He, Y.: Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '20*, pp. 3505–3506. Association for Computing Machinery, New York, NY, USA (2020). <https://doi.org/10.1145/3394486.3406703>. <https://doi.org/10.1145/3394486.3406703>
- [34] You, Y., Wang, Y., Zhang, H., Zhang, Z., Demmel, J., Hsieh, C.: The limit of the batch size. *CoRR* **abs/2006.08517** (2020) <https://arxiv.org/abs/2006.08517>
- [35] Yin, J., Tsaris, A., Dash, S., Miller, R., Wang, F., Shankar, M.A.: Comparative evaluation of deep learning workloads for leadership-class systems. *BenchCouncil Transactions on Benchmarks, Standards and Evaluations* **1**(1), 100005 (2021). <https://doi.org/10.1016/j.tbench.2021.100005>
- [36] Zhou, Q., Anthony, Q., Xu, L., Shafi, A., Abduljabbar, M., Subramoni, H., Panda, D.: Accelerating distributed deep learning training with compression assisted allgather and reduce-scatter communication. (2023)
- [37] Singh, S., Bhatele, A.: Exploiting Sparsity in Pruned Neural Networks to Optimize Large Model Training (2023)

- [38] Zheng, L., Li, Z., Zhang, H., Zhuang, Y., Chen, Z., Huang, Y., Wang, Y., Xu, Y., Zhuo, D., Xing, E.P., Gonzalez, J.E., Stoica, I.: Alpa: Automating inter- and Intra-Operator parallelism for distributed deep learning. In: 16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22), pp. 559–578. USENIX Association, Carlsbad, CA (2022). <https://www.usenix.org/conference/osdi22/presentation/zheng-lianmin>