NOVEMBER 2023 | Omer Dayan & Ekin Karabulut

A THROUGHPUT PERFORMANCE BENCHMARKING:

Pretraining NVIDIA NeMo GPT-3 on Kubernetes with Run:ai and Bare-metal

Abstract

This benchmarking study focuses on evaluating the distributed training throughput performance of GPT-3 models across different computing environments: "Bare-metal" and "Kubernetes with Run:ai." We conduct this benchmarking with a three-fold motivation: firstly, to compare throughput performance between the two environments; secondly, to explore how performance varies in a distributed training setting by investigating scenarios with 1, 2, and 4 nodes of an NVIDIA DGX BasePOD™ system; and thirdly, to understand how different model sizes of NeMo™ Megatron GPT-3 models impact performance.

For the GPT-3 model with 5 billion parameters, we find minimal differences in throughput between Kubernetes with Run:ai and bare-metal setups, ranging from 1.24% to 2.05%. Both setups exhibit similar linear scaling behavior, showcasing the robustness of Kubernetes with Run:ai for LLM training.

In contrast, for the GPT-3 model with 126 million parameters, Kubernetes with Run:ai demonstrates superior efficiency, resulting in a 10.27% improvement in throughput compared to bare-metal.

Our findings highlight the suitability of Kubernetes with Run:ai for scaling and optimizing performance, especially when dealing with models of varying sizes.

Kubernetes scripts shared in this study are accessible in our repository for researchers and developers who seek to harness the power of Kubernetes for distributed computing to train and optimize language models efficiently.

Table of contents

Introduction

In the rapidly evolving landscape of artificial intelligence, the performance and efficiency of training large-scale language models are of paramount importance. As the demand for sophisticated language understanding models continues to grow, the need for robust and efficient training infrastructure becomes increasingly critical. In this benchmarking whitepaper, we delve into a comprehensive comparison of the training throughput of GPT-3 models, utilizing two prominent and widely used orchestration platforms: Kubernetes with the integration of the Run:ai platform and bare-metal, where we utilize Slurm workload manager.

Our focus extends to evaluating two variants of NeMo™ Megatron's GPT-3 models: GPT-3 5B and GPT-3 126M. With their scale and complexity, training these models require cutting-edge hardware and optimized software frameworks to achieve optimal performance. Therefore, for our hardware infrastructure, we harnessed the power of an NVIDIA DGX BasePOD™ consisting of 4 NVIDIA DGX™ A100 nodes. This hardware configuration, known for its high-performance networking, was instrumental in achieving the training throughput we present in this study. NeMo™ Megatron, a creation of NVIDIA's Applied Deep Learning Research team, represents a GPU-accelerated framework tailored for training and deploying transformer-based Large Language Models (LLMs). We use it for its ability to handle models of up to a trillion parameters, offering a cost-effective and swift path to train generative AI.

At the heart of our investigation lies Kubernetes and Slurm. Slurm, an open-source workload manager and job scheduler, has gained adoption in high-performance computing (HPC) clusters due to its adaptability and resource allocation capabilities. On the other hand, Kubernetes, an open-source container orchestration platform, has emerged as a de facto standard for cloud-native infrastructure management and has become the platform of choice for AI companies like OpenAI and Spotify, and new AI Cloud providers like Coreweave. Its dynamic nature and cloud-native architecture make it a prime contender for orchestrating distributed machine learning workloads.

Our exploration would not be complete without the integration of the Run:ai platform, a pivotal component in our benchmarking endeavor on Kubernetes. Run:ai is an AI compute platform, enhances the Kubernetes ecosystem with advanced scheduling capabilities, cluster management, and GPU virtualization techniques for AI containerized workloads, thereby streamlining the training and deployment processes of AI models and maximizing availability and utilization of AI compute.

Throughout this whitepaper, we outline the infrastructure setup for both Kubernetes and bare-metal environments. We delve into the benchmarking methodologies, the experimental setup, and the results obtained from both environments. Notably, we highlight the NeMo-Megatron-Launcher, as well as the custom scripts sourced from the Run:ai k8s-launcher repository, adaptable for Kubernetes deployments —including standalone Kubernetes clusters. We make these scripts readily accessible in our repository, fostering an environment of collaborative exploration.

Data preprocessing

Dataset: The Pile
"The Pile" is a massive dataset that represents one of the largest publicly available collections of diverse natural language data. Compiled and released by OpenAI, "The Pile" is a culmination of various sources like books, articles, websites, and other written material from the internet. With the intention of promoting advances in natural language processing (NLP) and machine learning, the dataset spans an extensive range of topics, languages, and writing styles. It contains over hundreds of gigabytes of text data, making it a valuable resource for training state-of-the-art language models and developing AI systems capable of comprehending and generating human-like language. Its size and diversity enable researchers and developers to address a wide array of NLP challenges, leading to the advancement of various language-related applications across multiple domains. The Pile is provided as 30 shards of 15 GB size each. For our benchmark purposes we will download, extract and preprocess only 2 shards.

Data Locality
The preprocessed data is going to be available to all the nodes in the cluster through NFS server with Read/Write spec of approximately 500 GB/s. We preprocess the data such that the length of every sequence is 2048 tokens.

Cluster preparation

Run:ai and Kubernetes
Kubernetes, often abbreviated as K8s, is an open-source container orchestration platform designed to automate the deployment, scaling, and management of containerized applications. Originally developed by Google and now maintained by the Cloud Native Computing Foundation (CNCF), Kubernetes provides a powerful and flexible solution for running distributed applications in a cloud-native environment. With Kubernetes, developers can package their applications in containers, such as Docker, along with all their dependencies and configurations, ensuring consistent deployment across various environments.
Run:ai is installed on top of Kubernetes.

Bare-metal
For bare-metal experiments, we utilize Slurm. Slurm, short for "Simple Linux Utility for Resource Management," is a widely used open-source workload manager and job scheduler for high-performance computing (HPC) clusters. Developed and maintained by the Lawrence Livermore National Laboratory, Slurm manages and allocates computing resources, such as CPUs, GPUs, memory, and other hardware components, to different tasks and jobs submitted by users. It provides a robust and flexible platform for distributing and executing computational workloads in a multi-user environment.

In addition, we are going to use the Pyxis plugin to run enroots as described in the NeMo-Megatron-Launcher repository.

- NFS Server
- 4 x NVIDIA DGX A100 Nodes, with a total of 32 x NVIDIA A100 Tensor Core GPUs with 80 GB of GPU memory each
- 8 x 200 Gb HDR NVIDIA InfiniBand connectivity per node

Benchmarking setup

Models
We will run the benchmarking on NVIDIA NeMo™ GPT-3 models in the following sizes:

126M parameters
5B parameters

Run
For running on Kubernetes we will use specifically written scripts and tools that can be found in Run:ai k8s-launcher repository.

Throughput calculation
The throughput stands for the amount of tokens processed per second over the job's entire runtime. We measure the training throughput by the following equation based on step train time:

$$\text{Tokens per second} = \frac{\text{Global Batch Size} \times \text{Sequence Length}}{\text{Seconds per Step}}$$

Results

In our experiments, we aimed to assess the throughput performance of GPT-3 training across different computing environments, namely "Kubernetes with Run:ai" and "Bare-metal." Our investigation focused on three main objectives: comparing throughput performance of both environments, exploring the impact of distributed training settings with varying node configurations (1, 2, and 4 nodes), and evaluating the influence of different GPT-3 model sizes (5 billion parameters and 126 million parameters). In our results, we introduce the following metrics:

01. Tokens Per Second
Also called as throughput, this metric quantifies the rate at which the systems process tokens. It serves as a crucial indicator of processing efficiency.

02. Linear Factor
The Linear Factor is a relative measure of performance, indicating how each configuration compares to a baseline. A Linear Factor greater than 1 signifies improved performance, while a factor less than 1 suggests a decrease in performance compared to the baseline.

03. Perfect Linear Scale
The "Perfect Linear Scale" demonstrates the theoretical performance if scaling were perfectly linear.

04. The Difference (Diff.)
The "Diff." values represent the performance advantage or disadvantage of the "Bare-metal" setup compared to the "Kubernetes + Run:ai" setup for each configuration, expressed as a percentage. To calculate the percentage difference, we use the following formula:

$$\text{Difference} = 100 \times \frac{\text{Tokens per second(Bare Metal)} - \text{Tokens per second(Kubernetes with Run:ai)}}{\text{Tokens per second(Bare Metal)}}$$

GPT-3 - 5B parameters

		1 Node/ 8 GPUs	2 Nodes / 16 GPUs	4 Nodes / 32 GPUs
Bare-metal	Tokens Per Second	48,941	97,541	185,090
	Linear Factor / Perfect Linear Factor	1 / 1	1.99 / 2	3.78 / 4
	Perfect Linear Scale	48,941	97,882	195,764
Kubernetes + Run:ai	Tokens Per Second	48,131	95,545	182,791
	Linear Factor / Perfect Linear Factor	1 / 1	1.98 / 2	3.79 / 4
	Perfect Linear Scale	48,131	96,262	19,2524
	Diff.	**1.65%**	**2.05%**	**1.24%**

Table 1: Throughput Comparison of GPT-3 with 5B parameters.

Starting with the GPT-3 model with 5 billion parameters, we measured the rate at which tokens were processed per second for each setup and then interpolated these measurements. Detailed values can be found in Table 1, and visualizations are provided in Figure 1 and Figure 2. Our investigation revealed that there is a very slight difference in throughput between Kubernetes with Run:ai and bare-metal when using a large language model (LLM). Specifically, the difference amounted to just 1.65% and 2.05% for scenarios involving 1 and 2 nodes respectively, and 1.24% for 4 nodes. Additionally, both settings showed similar linear scaling behavior for each scenario. For instance, when the number of nodes doubled from 1 to 2, the bare-metal environment achieved a throughput factor of 1.99, whereas Kubernetes with Run:ai had a factor of 1.98. With 4 nodes, bare-metal exhibited a scaling factor of 3.78, and Kubernetes with Run:ai had a factor of 3.79.

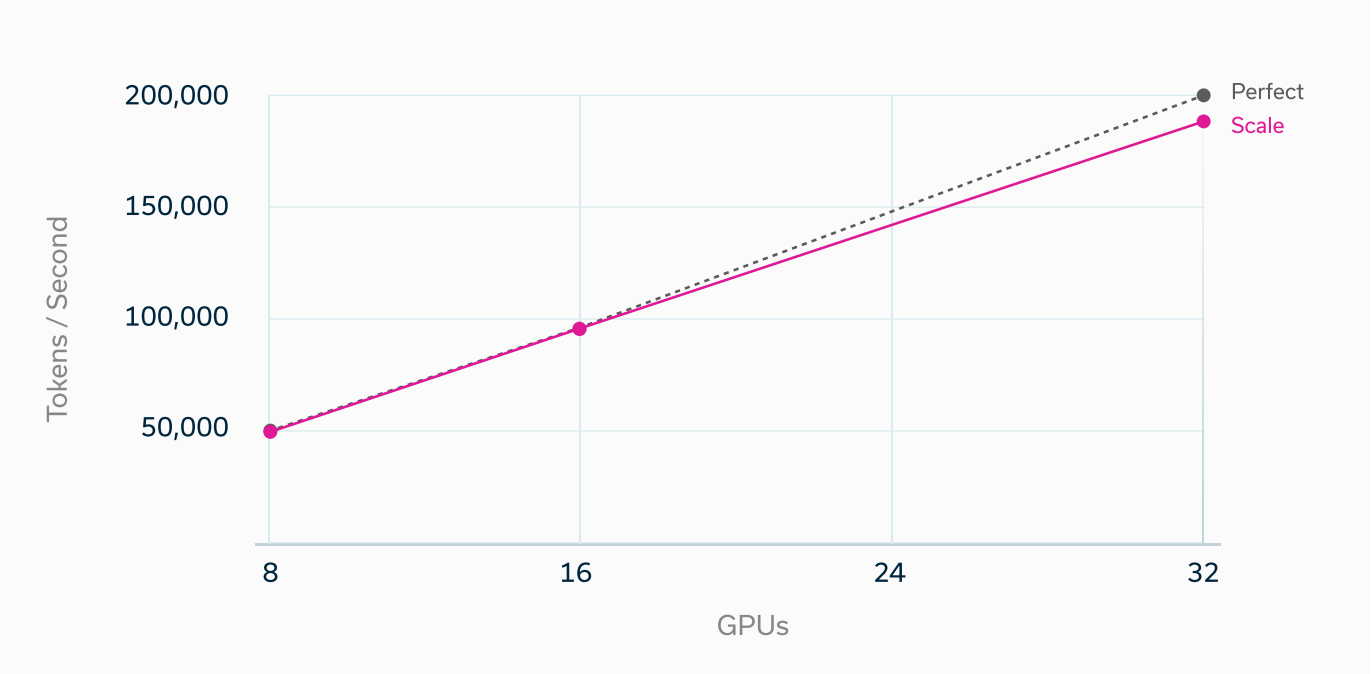

Figure 1: Tokens per seconds for 8 - 32 GPUs on bare-metal (GPT-3 5B parameters)

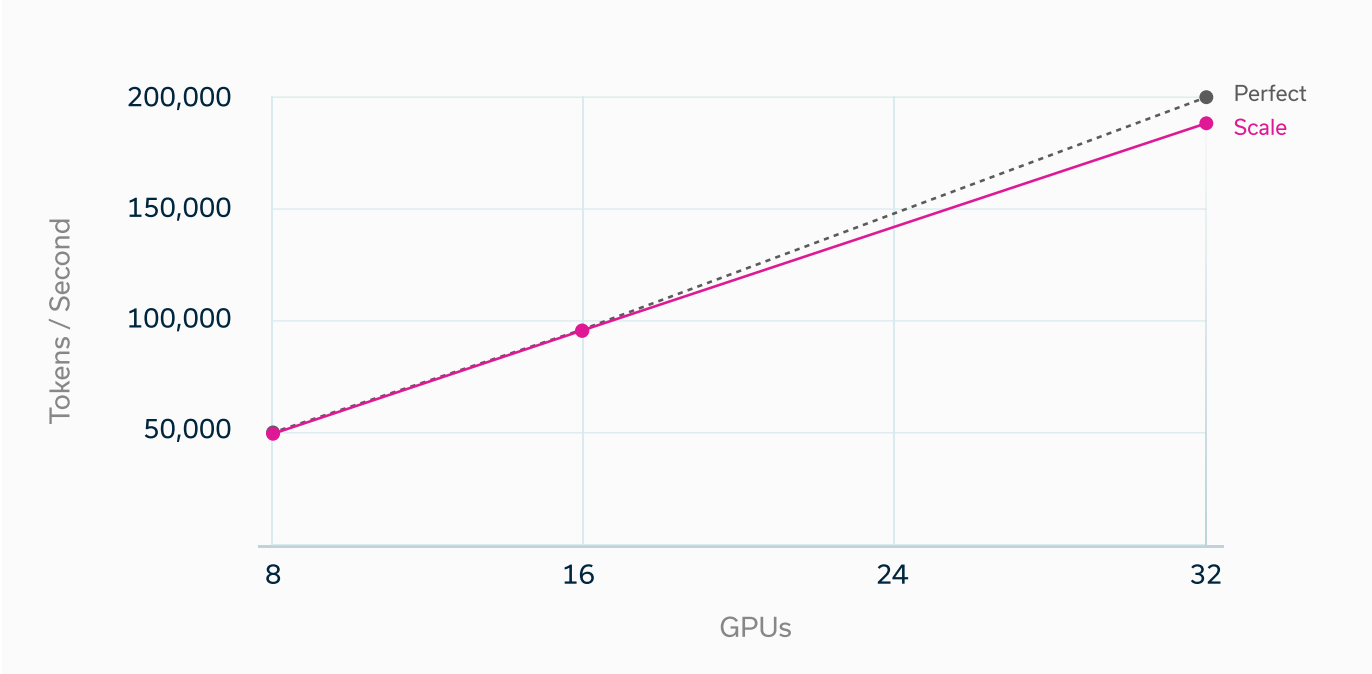

Figure 2: Tokens per seconds for 8 - 32 GPUs on Kubernetes with Run:ai (GPT-3 5B parameters)

GPT-3 - 126M Parameters

		1 Node/ 8 GPUs	2 Nodes / 16 GPUs	4 Nodes / 32 GPUs
Bare-metal	Tokens Per Second	919,803	1,747,626	3,236,345
	Linear Factor / Perfect Linear Factor	1 / 1	1.9 / 2	3.5 / 4
	Perfect Linear Scale	919,803	1,839,606	3,679,212
Kubernetes + Run:ai	Tokens Per Second	892,208	1,724,417	3,568,835
	Linear Factor / Perfect Linear Factor	1 / 1	1.93 / 2	3.65 / 4
	Perfect Linear Scale	894,208	1,784,416	3,268,832
	Diff.	**2.78%**	**1.33%**	**-1.00%**

Table 2: Throughput Comparison of GPT-3 with 126M parameters

Turning to the GPT-3 model with 126 million parameters, our subsequent experiments demonstrated nearly linear scalability in throughput when using Kubernetes with Run:ai **(see Table 2)**. The linear factor calculated for this scenario was 1.93 for 2 nodes and 4 for 4 nodes. With bare-metal, the corresponding factors were 1.9 and 3.5. Remarkably, when we analyzed the throughput difference between the two environments, we observed a 10.27% improvement with Kubernetes in comparison to bare-metal. From our observations, we can conclude that for GPT-3 with 5B parameters, the difference in throughput values between Kubernetes with Run:ai and bare-metal is not significant. However, for GPT-3 with 126M parameters, Kubernetes proves to be the more efficient choice for LLM training.

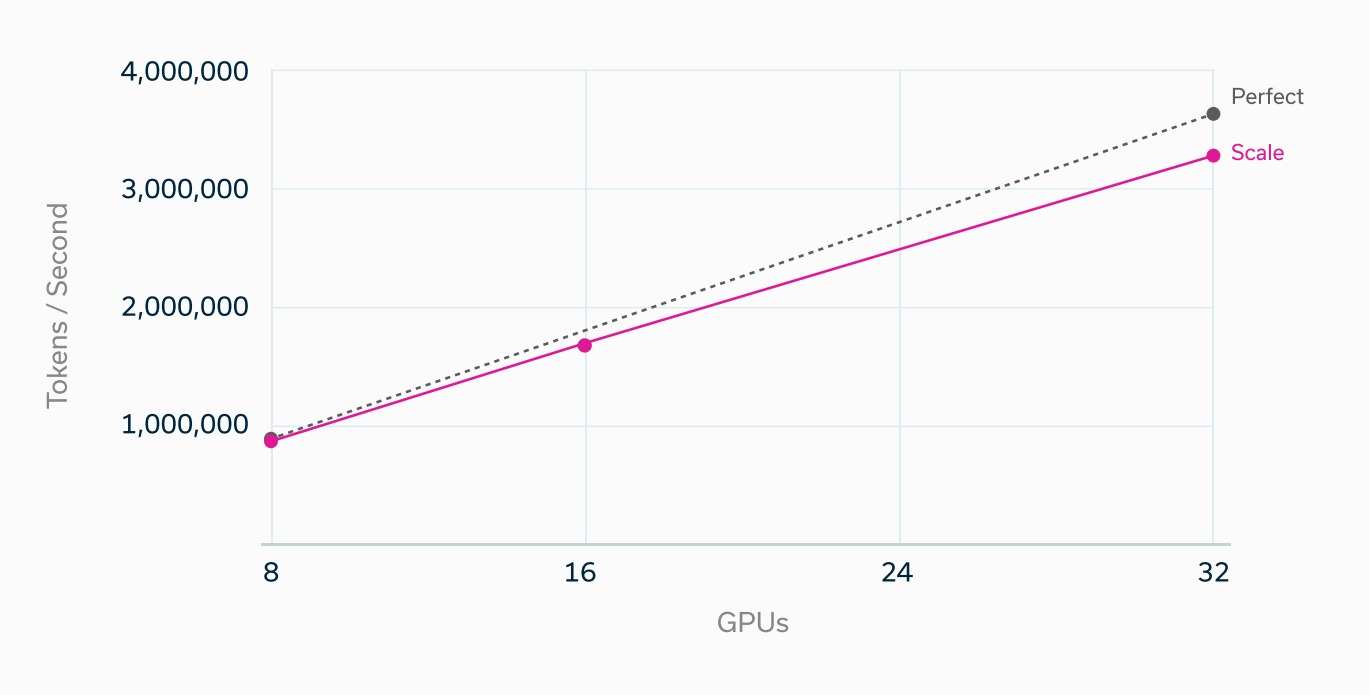

Figure 3: Tokens per seconds for 8 - 32 GPUs on bare-metal (GPT-3 126M parameters)

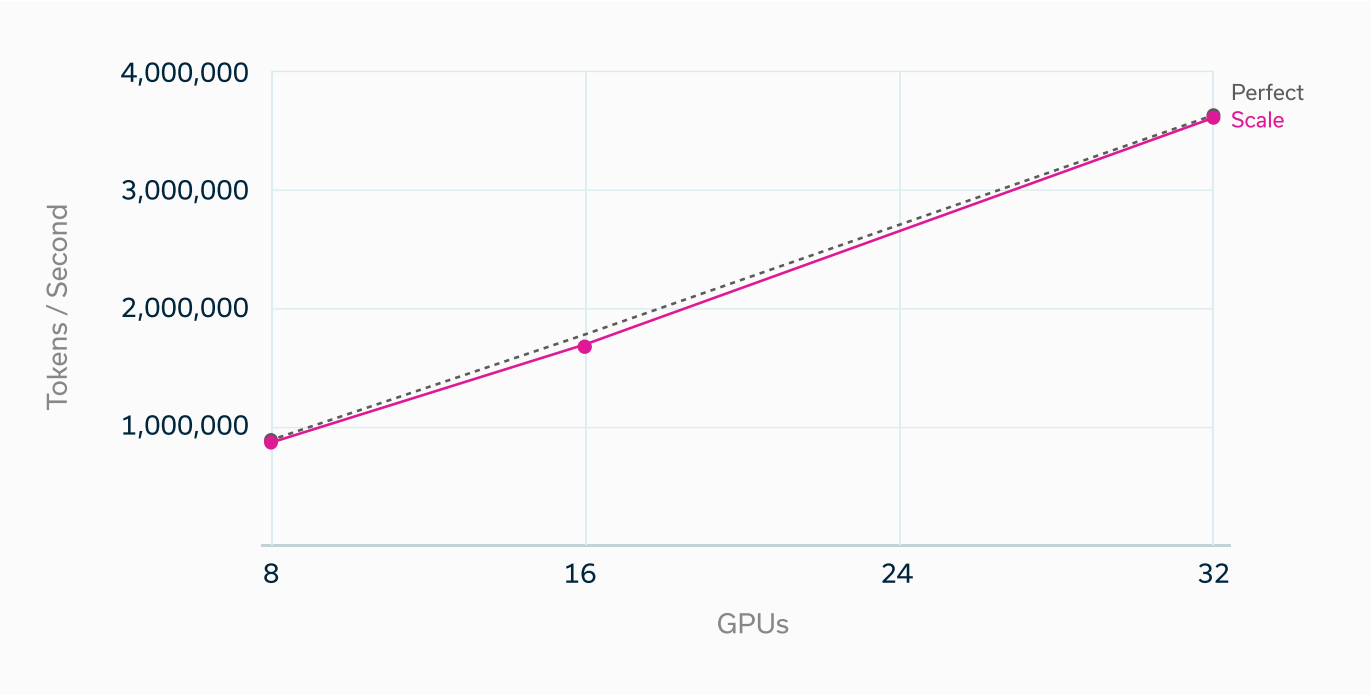

Figure 4: Tokens per seconds for 8 - 32 GPUs on Kubernetes with Run:ai (GPT-3 126M parameters)

Conclusion

Our experimentation revealed that for the GPT-3 model with 5 billion parameters, the difference in throughput between Kubernetes with Run:ai and bare-metal is minimal. The differences, amounting to around 1.65% for smaller node counts and 1.24% for larger ones, underscore the comparable performance of both platforms. Notably, the linear scaling factors observed in these cases remained consistent across the platforms, indicating a balanced scaling behavior.

For the GPT-3 model with 126 million parameters, the performance dynamics shifted. With a remarkable 10.27% improvement in throughput compared to bare-metal, Kubernetes affirm as an efficient choice for training large language models.

Taken together, our findings highlight the nuanced interplay between platform choice, model size, and scalability when aiming to achieve optimal training throughput. Researchers and practitioners can leverage these insights to make informed decisions when selecting the most suitable platform and configuration for their specific training needs.

Additionally, we're committed to ensuring the reproducibility of our experiments. As such, we've made our k8s launcher scripts available in our repository. This toolkit expands possibilities by providing a comprehensive suite of tools and scripts tailored for NVIDIA NeMo models. Designed to facilitate tasks from pretraining to fine-tuning and evaluation of expansive language models, this toolkit enhances the training experience for researchers and developers. By harnessing Kubernetes for distributed computing, we aim to simplify training processes and empower machine learning practitioners, fostering advancements in the field collectively.