Capability of ZeRO Engine

Eliminates memory redundancies

Train models with up to 13 billion parameters without requiring model parallelism

Partitioning optimizer states

Significantly increasing model size and performance compared to state-of-the-art.

Used to create the world’s largest language model

Data parallelism

Parallel Processing

Data Distribution

Simultaneous Execution

Aggregation

Scalability

Load Balancing

Display price: per hour per month

CPU basic
2 vCPU · 16 GB RAM
Current · Free

CPU upgrade
8 vCPU · 32 GB RAM
$0.03/hour

Nvidia T4 small
4 vCPU · 15 GB RAM · 16GB VRAM
$0.60/hour

Nvidia T4 medium
8 vCPU · 30 GB RAM · 16GB VRAM
$0.90/hour

Nvidia A10G small
4 vCPU · 15 GB RAM · 24GB VRAM
$1.05/hour

Nvidia A10G large
12 vCPU · 46 GB RAM · 24GB VRAM
$3.15/hour

Nvidia A100 large
12 vCPU · 142 GB RAM · 40GB VRAM
$4.13/hour

AI Accelerator
HPU · IPU · ...
Coming soon

PPO

Policy Optimization Problem

Advantage Estimation

Objective Function

Value Function

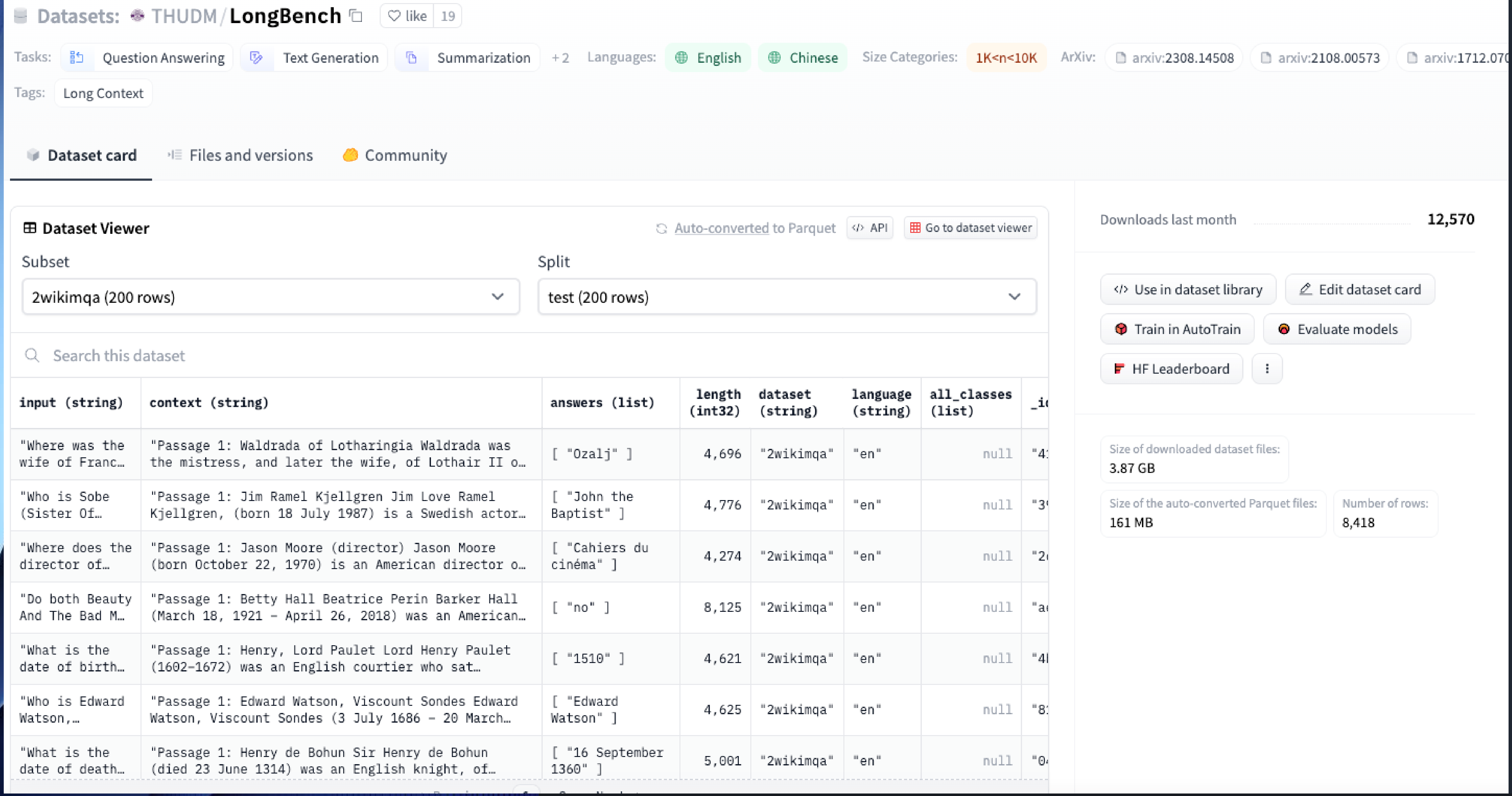
Clipped Surrogate Objective

Policy Updates

Multiple Epochs

LLMs Training

Collecting and Processing Data


Datasets: THUDM / LongBench like 19

Tasks: Question Answering Text Generation Summarization +2 Languages: English Chinese Size Categories: 1K<n<10K ArXiv: arxiv:2308.14508 arxiv:2108.00573 arxiv:1712.070

Tags: Long Context

Dataset card | Files and versions | Community

Dataset Viewer Auto-converted to Parquet | API | Go to dataset viewer

Subset: 2wikimqa (200 rows)

Split: test (200 rows)

Search this dataset

input (string)	context (string)	answers (list)	length (int32)	dataset (string)	language (string)	all_classes (list)	_id
"Where was the wife of Franc…	"Passage 1: Waldrada of Lotharingia Waldrada was the mistress, and later the wife, of Lothair II o…	["Ozalj"]	4,696	"2wikimqa"	"en"	null	"4
"Who is Sobe (Sister Of…	"Passage 1: Jim Ramel Kjellgren Jim Love Ramel Kjellgren, (born 18 July 1987) is a Swedish actor…	["John the Baptist"]	4,776	"2wikimqa"	"en"	null	"3
"Where does the director of…	"Passage 1: Jason Moore (director) Jason Moore (born October 22, 1970) is an American director o…	["Cahiers du cinéma"]	4,274	"2wikimqa"	"en"	null	"2
"Do both Beauty And The Bad M…	"Passage 1: Betty Hall Beatrice Perin Barker Hall (March 18, 1921 – April 26, 2018) was an American…	["no"]	8,125	"2wikimqa"	"en"	null	"a
"What is the date of birth…	"Passage 1: Henry, Lord Paulet Lord Henry Paulet (1602–1672) was an English courtier who sat…	["1510"]	4,621	"2wikimqa"	"en"	null	"4
"Who is Edward Watson,…	"Passage 1: Edward Watson, Viscount Sondes Edward Watson, Viscount Sondes (3 July 1686 – 20 March…	["Edward Watson"]	4,625	"2wikimqa"	"en"	null	"8
"What is the date of death…	"Passage 1: Henry de Bohun Sir Henry de Bohun (died 23 June 1314) was an English knight, of…	["16 September 1360"]	5,001	"2wikimqa"	"en"	null	"0

Downloads last month 12,570

Use in dataset library | Edit dataset card | Train in AutoTrain | Evaluate models | HF Leaderboard

Size of downloaded dataset files: 3.87 GB

Size of the auto-converted Parquet files: 161 MB

Number of rows: 8,418

LLMs Training

Tokenization

- Tokenization is the process of breaking down a sequence of text into smaller units, known as tokens.
- Tokens can be words, subwords, characters, or other linguistic units depending on the chosen tokenization strategy.
- Tokens can also be the “label” of the task when the LLMs try to figure out what the users want.

LLMs Training

Tokenization Strategies

- Word Tokenization: Breaking text into individual words
- Subword Tokenization: Breaking down words into smaller subword units.
- Character Tokenization: Treating each character as a separate token.
- Byte-Pair Encoding (BPE): A subword tokenization technique that identifies the most frequent pairs of characters and replaces them with a special token, iteratively building up a vocabulary of subwords.
- Sentence Tokenization: Breaking a text into individual sentences.

DeepSpeed Chat

DeepSpeed Chat is a powerful AI training system that utilizes cutting-edge technology to create advanced dialogue systems. In this presentation, we will explore the principles behind this innovative system and its many applications.

by Binyuan

LLMs Training

Choose the Right Framework and Programming Language

- Choosing the right framework and programming language is crucial for LLMs training, such as PyTorch, TensorFlow, and Keras, and programming languages like Python and Julia.

Collecting and Processing Data

- Data collection and preparation is an essential step in LLMs training. Gather a diverse and extensive dataset of text. This dataset will be used to train the model. The larger and more varied the dataset, the better your model will likely be.
- Clean and preprocess the text data. This involves tasks like removing special characters, converting text to lowercase, tokenization (splitting text into words or subwords), and more.

Tokenization, Vocabulary, Model Architecture, and Implementations

- Tokenize the text into smaller units, such as words or subwords. Build a vocabulary from these tokens and assign each token a unique numerical identifier. (words - token)
- Choose a deep learning architecture for your language model. Transformer-based architectures, like the one used in GPT models, are currently state-of-the-art for natural language processing tasks.
- Implement the chosen model architecture using the selected framework. This involves creating layers, attention mechanisms, and other components specific to your chosen architecture.

LLMs Training

Training & Fine-Tuning

- Train your model on the preprocessed dataset. Training a language model like GPT from scratch requires a significant amount of computational power, often utilizing multiple GPUs or even TPUs. During training, the model learns to predict the next word in a sentence given the previous words.
- After initial training, you might fine-tune the model on a narrower dataset or specific task to make it more relevant for a particular use case.

Evaluation

- Evaluate the performance of your model using various metrics such as perplexity, BLEU score, or more task-specific measures depending on your use case.

Deployment & Iterative Improvement

- Once your model is trained and evaluated satisfactorily, you can deploy it to serve user queries. This might involve setting up an API or integration into an application.
- Continuously improve your model by fine-tuning on new data, updating the architecture, or incorporating new techniques as the field evolves.

Steps within fine-tuning

Supervised Finetuning (SFT)

Human responses to various queries are carefully selected to finetune the pre-trained language models.

Reward Finetuning (RF)

A separate (usually smaller than the SFT) model (RW) is trained with a dataset that has human-provided rankings of multiple answers to the same query.

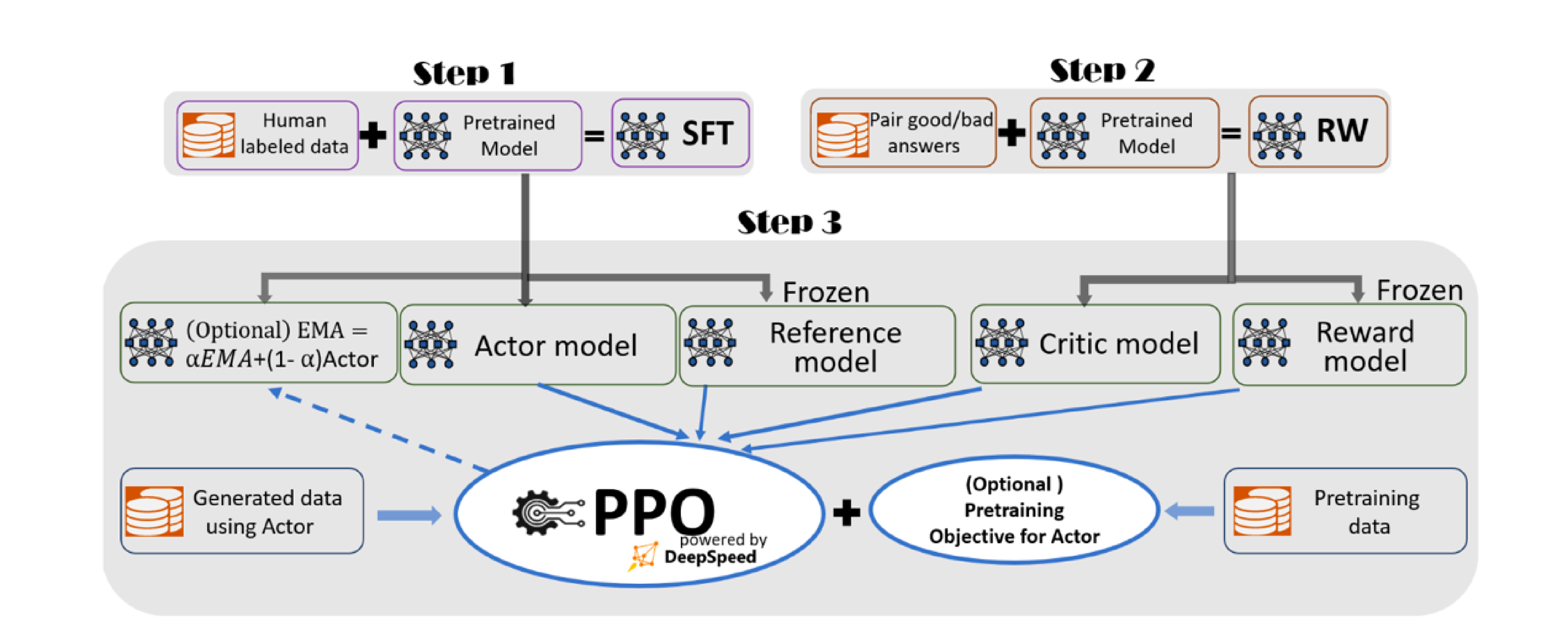
Reinforcement learning from human feedback (RLHF)

The SFT model is further finetuned with the reward feedback from the RW model using the Proximal Policy Optimization (PPO) algorithm.

Exponential Moving Average (EMA)

Mixture Training

Steps within fine-tuning



What is the Reinforcement Learning from Human Feedback (RLHF)?

What is the Reinforcement Learning from Human Feedback (RLHF)?

Reinforcement Learning from Human Feedback (RLHF) is an approach in machine learning where an agent learns to perform a task through a combination of trial and error and guidance from human feedback.

Work Flow of RLHF

1 Initial Policy

Set rules before start

The agents’ strategy for taking actions

2 Interaction with Environment

Taking action based on the current rules

Get feedback from the environment

(Remember the RW model?)

3 Human Feedback

Humans provide feedback that helps the agent understand the quality of its actions beyond the immediate rewards

Work Flow of RLHF

Policy Improvement

The agent uses the combination of the environment's rewards and the reward model derived from human feedback to update its policy.

4

5

6

Reward Model from Human Feedback

Set rules before start

The agents' strategy for taking actions

Iterative Process & Human in the loop

What is DeepSpeed Chat?

DeepSpeed is a deep learning optimization library that makes distributed training and inference easy, efficient, and effective.

A AI training System

DeepSpeed is a deep learning optimization library that makes distributed training and inference easy, efficient, and effective.

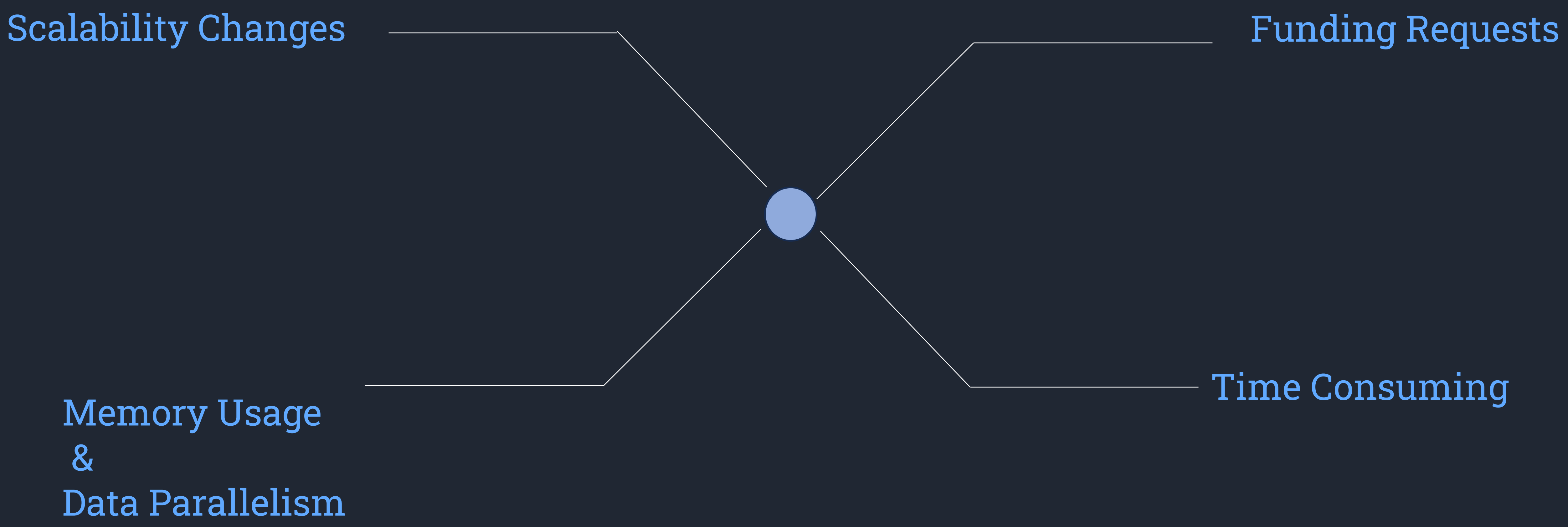
High speed, Low cost

Use less time and requires fewer GPUs when training language models

Intelligent in inference

DeepSpeed brings together innovations in parallelism technologies and combines them with high-performance custom inference kernels.

Major Challenge & Cost



Why So Advance?

Hybrid Engine Applied

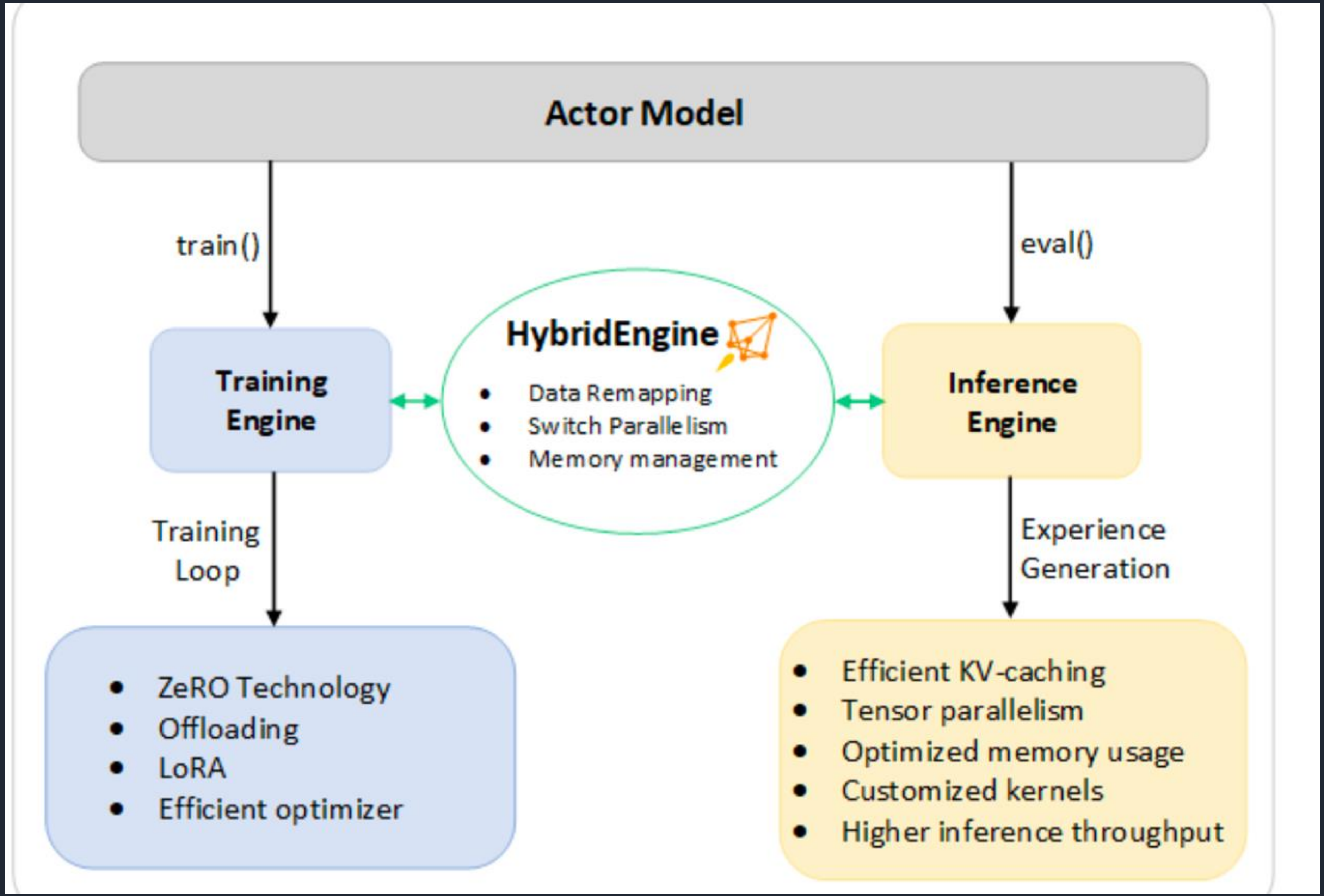
- DeepSpeed Chat applies a hybrid engine which can efficiently manage training engine and inference engine.
- Seamless transition between DeepSpeed training and inference is achieved with evaluation and training modes enabled for actor model.

Memory management

•Memory management, optimized kernels, and tensor parallelism boost inference throughput during experience generation phase.
•Memory optimization techniques like ZeRO and Low Rank Adaption (LoRA) are used during training execution.

Configure between different models

Memory system is reconfigured to maximize availability during different modes, avoiding bottlenecks and supporting large batch sizes.


Actor Model
train()
eval()
HybridEngine
Data Remapping
Switch Parallelism
Memory management
Training Engine
Inference Engine
Training Loop
Experience Generation
ZeRO Technology
Offloading
LoRA
Efficient optimizer
Efficient KV-caching
Tensor parallelism
Optimized memory usage
Customized kernels
Higher inference throughput

Intro to ZeRO++ Engine

ZeRO

ZeRO ++

- Reduces the communication volume
- Blocked quantized weights, hierarchical weight partitioning, and quantized gradient communication.
- Overlaps compute and communication, and uses optimized CUDA kernels for quantization, dequantization, and tensor slice reordering.

ZeRO-DP

- Using [Data parallelism](#) strategy
- Eliminates memory inefficiency in data parallelism

ZeRO-R

- Optimizes activation memory by identifying and removing activation replication in existing MP approaches
- ZeRO-R defines appropriate size for temporary buffers to strike for a balance of memory and computation efficiency.

ZeRO VS ZeRO ++

All-to-all quantized gradient reduction collective (qgZ)

This technique reduces gradient communication by 75% compared to reduce-scatter.

Blocked quantized weights (qwZ):

This technique reduces the communication volume of all-gather of weights by 50%.

Optimized integration

ZeRO++ optimally integrates each of the above techniques into the existing ZeRO implementation, translating the 4x communication volume reduction into real throughput improvement.

Hierarchical partitioning of model weights (hpZ)

This technique completely eliminates inter-node all-gather communication in backward propagation.

Capability

1 One step for multi training Process

- A single script capable of taking a pre-trained Huggingface model, running it through all three steps of InstructGPT training using the DeepSpeed-RLHF system, and producing your very own ChatGPT-like model.
- Combination of SFT, Reward Model, RLHF.
- Hybrid Engine with training and inferencing.

2 Cheap, Fast, and Heavy Duty

- DeepSpeed-HE is over 15x faster than existing systems, making RLHF training both fast and affordable.
- DeepSpeed-HE supports models with hundreds of billions of parameters and can achieve excellent scalability on multi-node multi-GPU systems.

3 User Modified Engine

- See The Markdown File

GPUs	OPT-6.7B	OPT-13B	OPT-30B	OPT-66B
8x A100-40GB	5.7 hours	10.8 hours	1.85 days	NA
8x A100-80GB	4.1 hours ($132)	9 hours ($290)	18 hours ($580)	2.1 days ($1620)

Single-Node 8x A100: Training Time and Corresponding Approximate Cost on Azure.

GPUs	OPT-13B	OPT-30B	OPT-66B	OPT-175B
64x A100-80G	1.25 hours ($320)	4 hours ($1024)	7.5 hours ($1920)	20 hours ($5120)

Multi-Node 64x A100-80GB: Training Time and Corresponding Approximate Cost on Azure

Model Sizes	Step 1	Step 2	Step 3	Total
Actor: OPT-1.3B, Reward: OPT-350M	2900 secs	670 secs	1.2hr	2.2hr

E2E time breakdown for training a 1.3 billion parameter ChatGPT model via DeepSpeed-Chat on a single commodity NVIDIA A6000 GPU with 48GB memory.

How to Deploy?

Questions and Answers

THANK YOU!

Ref

1. Yao Z, Aminabadi RY, Ruwase O, et al. DeepSpeed-Chat: Easy, Fast and Affordable RLHF Training of ChatGPT-like Models at All Scales. Published online August 2, 2023. Accessed August 21, 2023. http://arxiv.org/abs/2308.01320
2. ZeRO & DeepSpeed: New system optimizations enable training models with over 100 billion parameters - Microsoft Research. Accessed August 21, 2023. https://www.microsoft.com/en-us/research/blog/zero-deepspeed-new-system-optimizations-enable-training-models-with-over-100-billion-parameters/
3. Wang G, Qin H, Jacobs SA, et al. ZeRO++: Extremely Efficient Collective Communication for Giant Model Training. Published online June 16, 2023. Accessed August 21, 2023. http://arxiv.org/abs/2306.10209
4. Accelerate Large Model Training using DeepSpeed. Accessed August 21, 2023. https://huggingface.co/blog/accelerate-deepspeed
5. DeepSpeed/blogs/deepspeed-chat/README.md at master · microsoft/DeepSpeed. GitHub. Accessed August 21, 2023. https://github.com/microsoft/DeepSpeed/blob/master/blogs/deepspeed-chat/README.md