Training LLMs from scratch

How we built MPT-7B and MPT-30B

Abhinav Venigalla

Agenda

- Compute and Orchestration
  - Compute requirements
  - MosaicML Cloud
  - Determinism
  - Job failures and graceful resumption
- Training Runtime
  - Streaming Datasets
  - Composer
  - Fully Sharded Data Parallelism (FSDP)
  - LLM Foundry
- MPT Models
  - Data
  - Model Architecture, Pretraining
  - Finetuning
  - Eval

Compute + Orchestration

Compute requirements for LLMs

- Building LLMs from scratch takes a **LOT** of compute
- To finish training in human friendly time scales, we need 100s–1000s of GPUs
- Need tools for launching, resuming, managing runs on large GPU clusters

LLM Training Costs on MosaicML Cloud			
Model	Billions of Tokens (Compute-optimal)	Days to Train on MosaicML Cloud	Approx. Cost on MosaicML Cloud
GPT-1.3B	26B	0.14	$2,000
GPT-2.7B	54B	0.48	$6,000
GPT-6.7B	134B	2.32	$30,000
GPT-13B	260B	7.43	$100,000
GPT-30B *	**610B**	**35.98**	**$450,000**
GPT-70B **	1400B	176.55	$2,500,000

^ all using 256xA100

MosaicML Cloud

- **MosaicML Cloud** is a job orchestration + scheduling layer that sits on top of any compute cluster
- **Compute-agnostic:** run jobs on any cloud provider. You can rent compute directly from us, or run in your private VPC.
- **ML-specific:** features like scaling, resumption, object stores, experiment trackers are tailored for ML engineers
- **High performance and efficient!**

Multi-node Orchestration

```yaml
# chinchilla-13b-test.yaml

image: mosaicml/composer:latest

integrations:
  - integration_type: git_repo
    git_repo: mosaicml/benchmarks
    git_branch: main

command: |
  cd benchmarks/llm
  composer main.py /mnt/config/parameters.yaml

name: chinchilla-13b-test

cluster: r0z0
gpus: 256 # or 8, 16, 32, etc.

parameters:
  ...
```

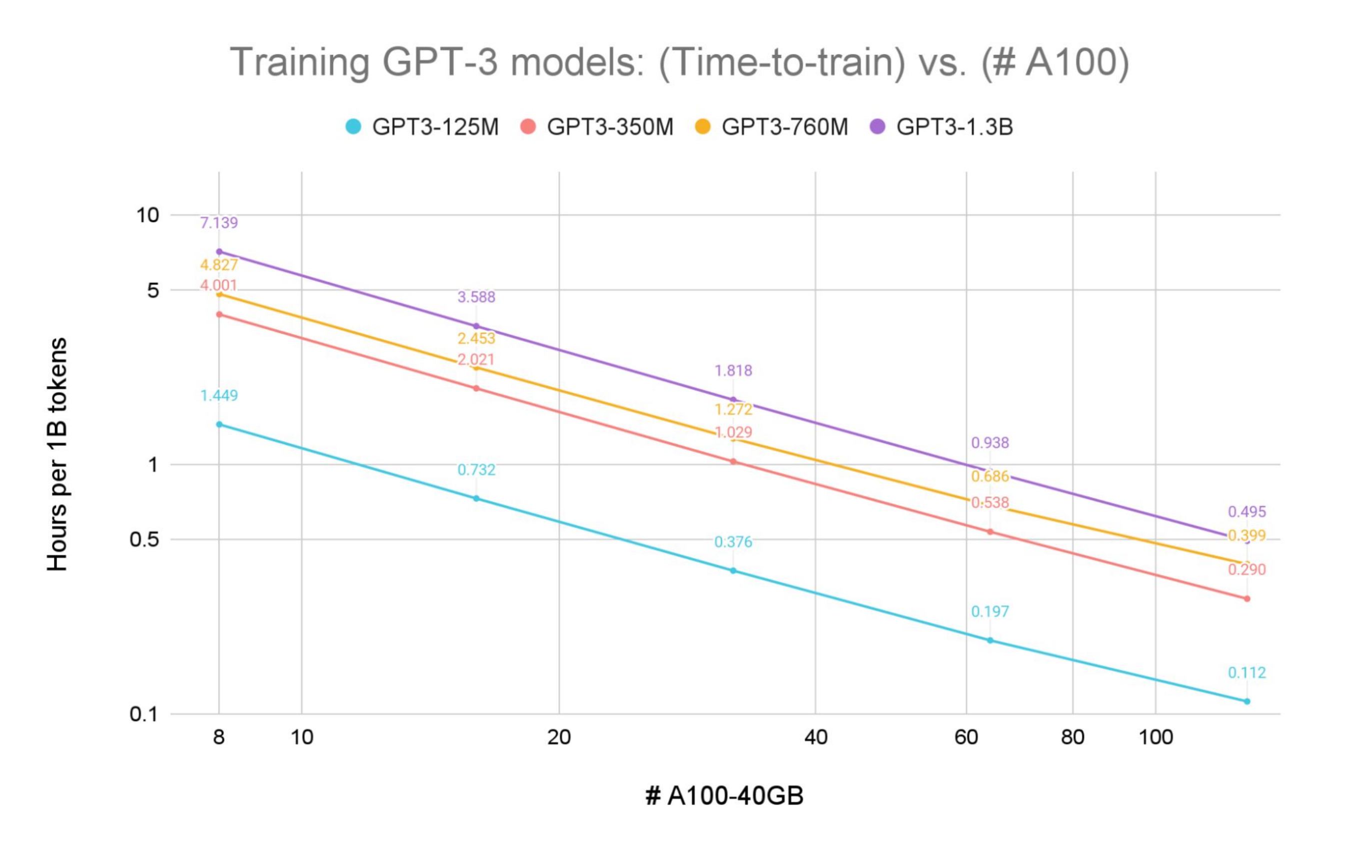
```
~ ▶ mcli run -f chinchilla-13b-test.yaml
------------------------------------------------------------
Let's run this run
------------------------------------------------------------

✔ Run chinchilla-13b-test-dgrj submitted.

To see the run's status, use:

mcli get runs

To see the run's logs, use:

mcli logs chinchilla-13b-test-dgrj
```

```
Active Runs:
NAME                              USER  AGE  NODE_NAME          GPUS_USED
chinchilla-13b-gpus-256-run1-5k7a abhi  6hr  a100-40gb-c8zmw    8
                                             a100-40gb-kyehy    8
                                             a100-40gb-dc9tv    8
                                             a100-40gb-emzgp    8
                                             a100-40gb-u4oyt    8
                                             a100-40gb-vwgy1    8
                                             a100-40gb-eimqi    8
                                             a100-40gb-jwrhu    8
                                             a100-40gb-fna6f    8
                                             a100-40gb-xd96v    8
                                             a100-40gb-fxvtt    8
                                             a100-40gb-oqu7a    8
                                             a100-40gb-l9oab    8
                                             a100-40gb-rklge    8
                                             a100-40gb-eiar1    8
                                             a100-40gb-l95rs    8
                                             a100-40gb-zfo3m    8
                                             a100-40gb-wmstz    8
                                             a100-40gb-kxwtn    8
                                             a100-40gb-x5hfv    8
                                             a100-40gb-3urlg    8
                                             a100-40gb-hlirp    8
                                             a100-40gb-xksr1    8
                                             a100-40gb-gb8mp    8
                                             a100-40gb-yc2d9    8
                                             a100-40gb-es31q    8
                                             a100-40gb-vy00v    8
                                             a100-40gb-8wjql    8
                                             a100-40gb-qyhsa    8
                                             a100-40gb-ffgro    8
                                             a100-40gb-czx36    8
                                             a100-40gb-z0nuh    8
```

That's a single job training GPT-13B on 256 GPUs!

Scaling from 1 → 8 → 256 → … GPUs is as easy as changing `gpus` at launch time.

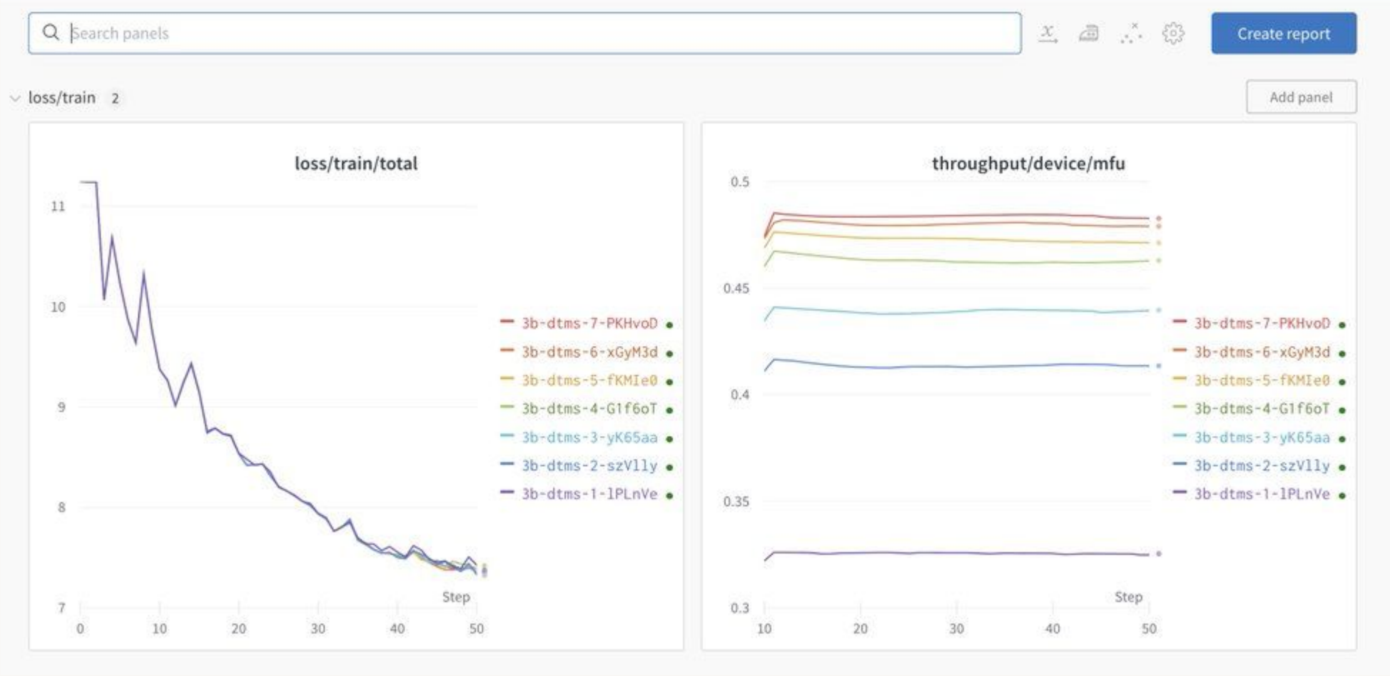
Multinode Scaling



Training the same GPT model + same batch size on 8 GPUs vs. 128 GPUs, we saw a 14.4x speedup (16x ideal)

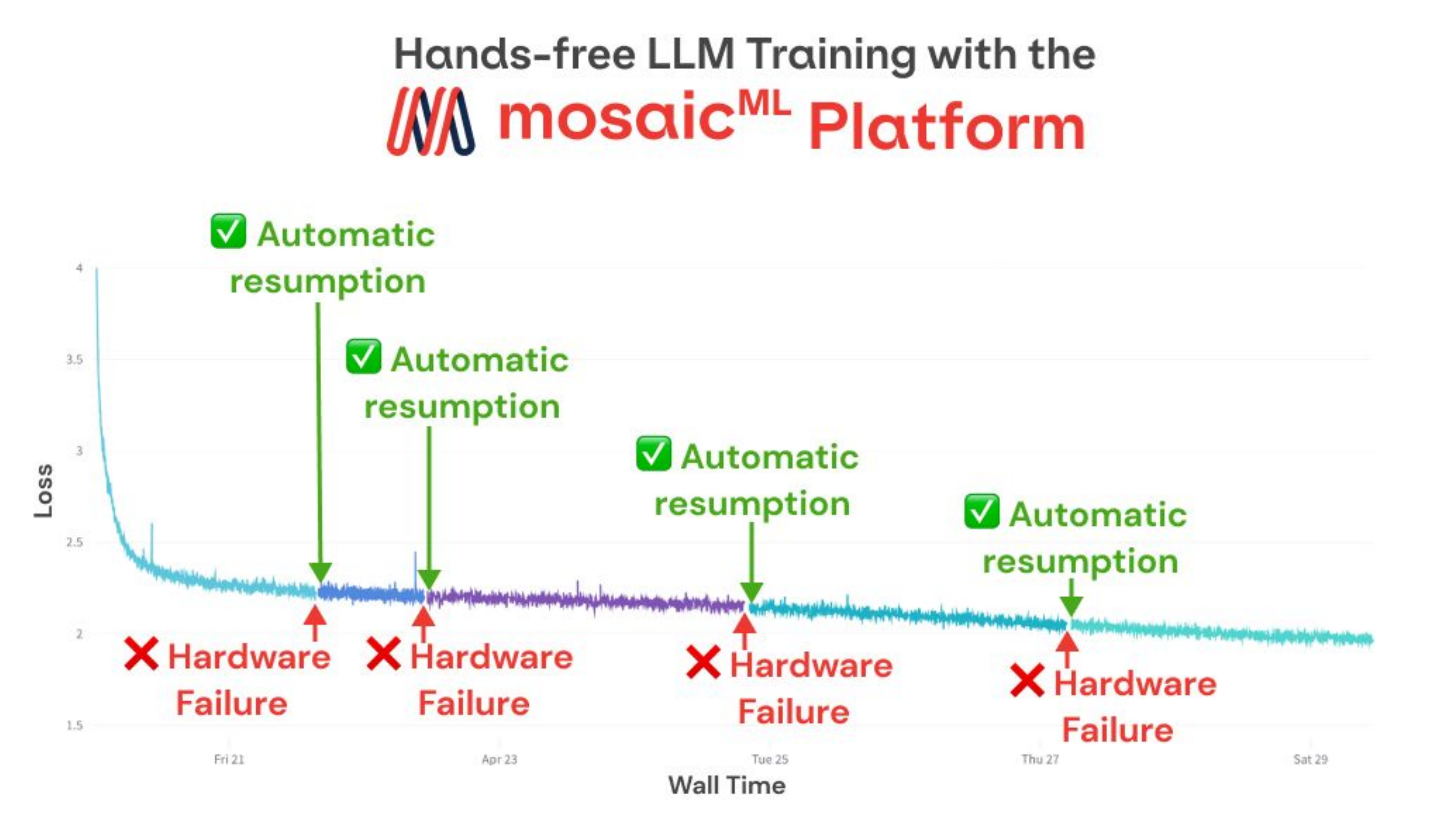
When you use fast inter-node networking, you get near-linear scaling with more GPUs. Your jobs get done faster, with only a minor increase in cost.

Determinism



To reproduce, debug, and improve our training recipes, we use a deterministic dataloader (left) and microbatching engine (right).

Graceful Resumption

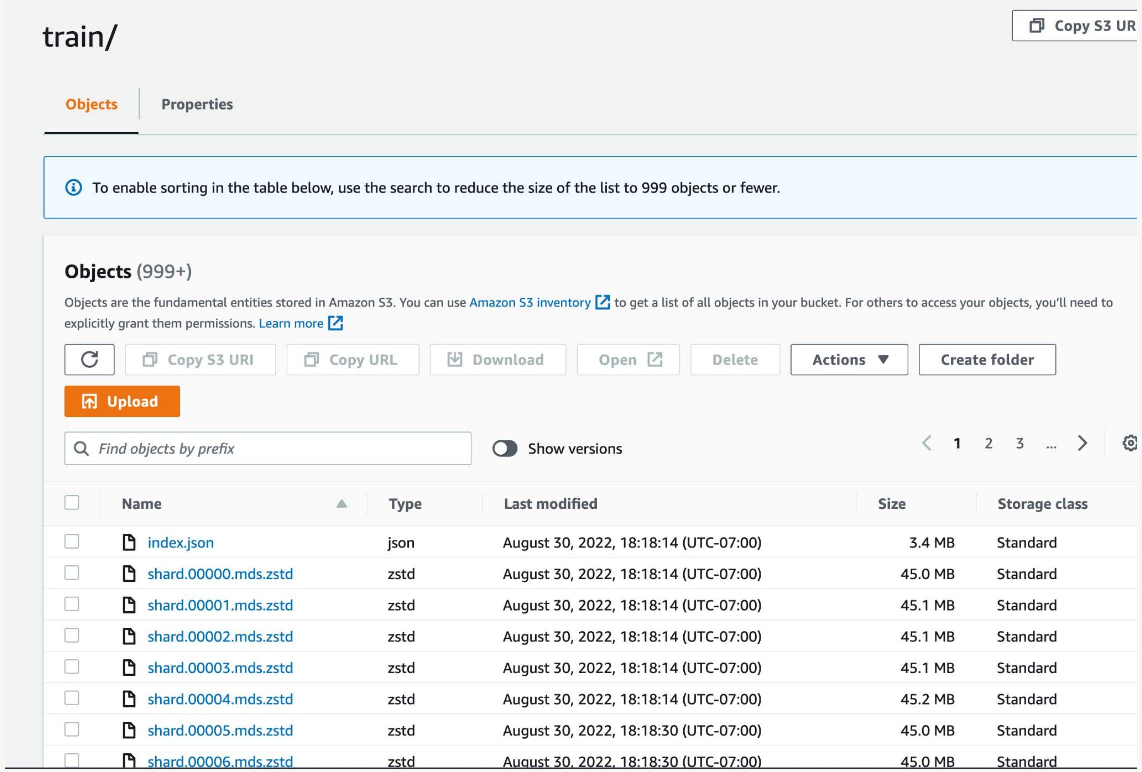


Hardware failures are common. Instead of researchers 'babysitting' these runs, our platform gracefully stops + resumes training

Training Runtime

Streaming Datasets

https://github.com/mosaicml/streaming



We stream data directly from cloud object stores (S3, GCS, OCI, R2) and stream checkpoints directly back. ML engineers get to work with a single source of ground truth, and IT gets to manage data security policies .

Composer

- Composer is a **PyTorch library** built for efficient ML training

- Takes care of details like mixed precision, distributed training, checkpointing, etc.

- Includes a 2-way callback system that allows users to **write and apply algorithms** during training.

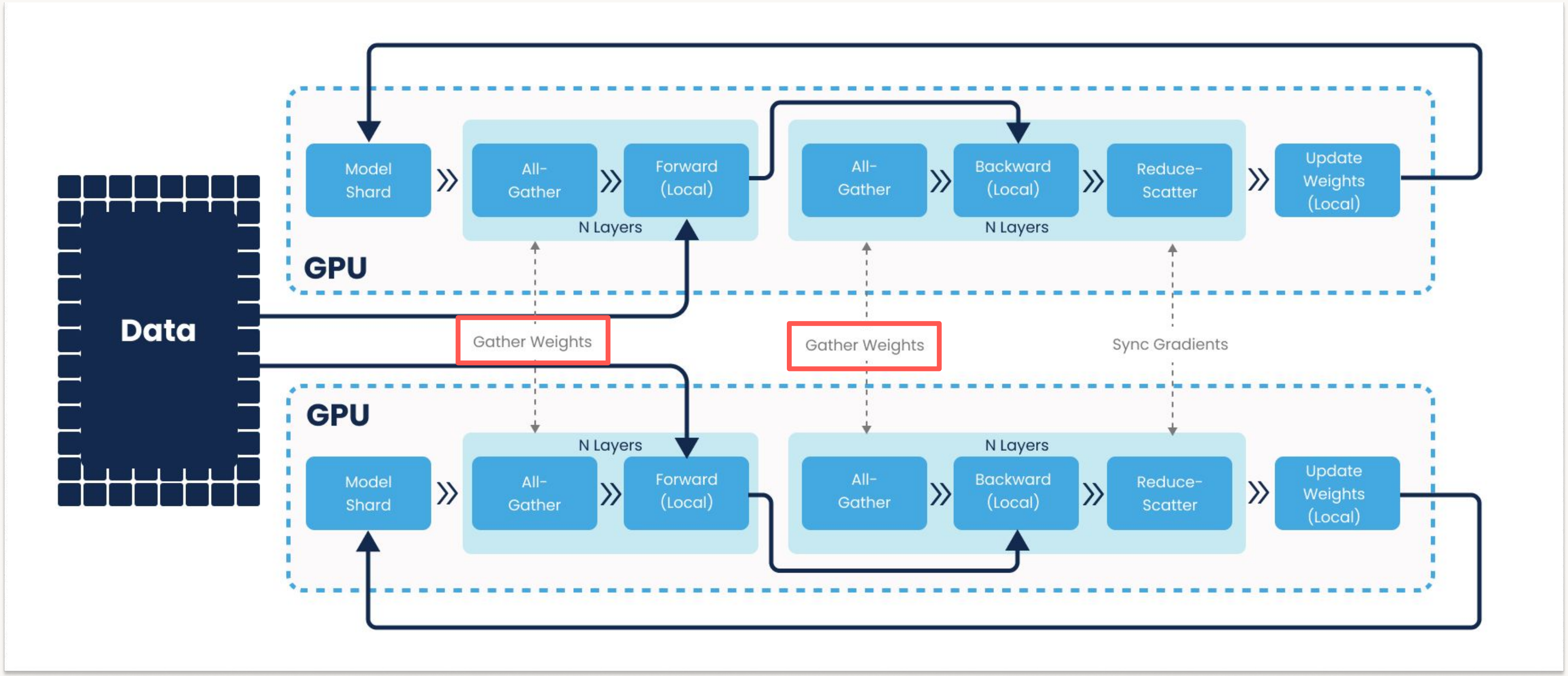
https://github.com/mosaicml/composer

```
from composer import Trainer
from composer import algorithms as algos

trainer = Trainer(
    algorithms=[
        algos.BlurPool(),
        algos.ChannelsLast(),
        algos.EMA(update_interval="20ba"),
        algos.LabelSmoothing(smoothing=0.08),
        algos.ProgressiveResizing(size_increment=4, delay_fraction=0.4)
    ],
    model=...,  # the torchvision resnet-50
    train_dataloader=...,  # use FFCV dataloaders
    …
)
```

Fully Sharded Data Parallelism (FSDP)

- PyTorch FullyShardedDataParallel (FSDP) is an execution strategy for training large models
- FSDP does the exact same math as **data-parallelism,** the only difference is the **storage location** for model+optimizer weights
- Your weights only need to **fit across the total cluster memory**, not on each GPU
- Super flexible: No model-, pipeline-, or tensor-parallelism required!
- **Composer has built-in FSDP support**

Fully Sharded Data Parallelism (FSDP)



FSDP shards the model + optimizer weights across all GPUs.
During each training step, individual layer weights are gathered when needed, and discarded when not. This saves tons of memory!

LLM Foundry

https://github.com/mosaicml/llm-foundry

python 3.8 | 3.9 | 3.10 pypi v0.2.0 slack chat License Apache 2.0

LLM Foundry

This repository contains code for training, finetuning, evaluating, and deploying LLMs for inference with Composer and the MosaicML platform. Designed to be easy-to-use, efficient *and* flexible, this codebase is designed to enable rapid experimentation with the latest techniques.

You'll find in this repo:

- `llmfoundry/` - source code for models, datasets, callbacks, utilities, etc.
- `scripts/` - scripts to run LLM workloads
  - `data_prep/` - convert text data from original sources to StreamingDataset format
  - `train/` - train or finetune HuggingFace and MPT models from 125M - 70B parameters
    - `train/benchmarking` - profile training throughput and MFU
  - `inference/` - convert models to HuggingFace or ONNX format, and generate responses
    - `inference/benchmarking` - profile inference latency and throughput
  - `eval/` - evaluate LLMs on academic (or custom) in-context-learning tasks
- `mcli/` - launch any of these workloads using MCLI and the MosaicML platform
- `TUTORIAL.md` - a deeper dive into the repo, example workflows, and FAQs

LLM Foundry is a complete toolkit for data prep, training, finetuning, evaluation, and inference. MPT-7B and MPT-30B were both built with LLM Foundry!

MPT Models

Data

- We used a variety of pretraining data sources to fill our 1T token budget
- Filtering, deduplication is crucial
- Picking proportions is important. E.g. trading off English vs. Code data
- Tokenizer design can impact some tasks a lot, e.g. math, code, foreign languages

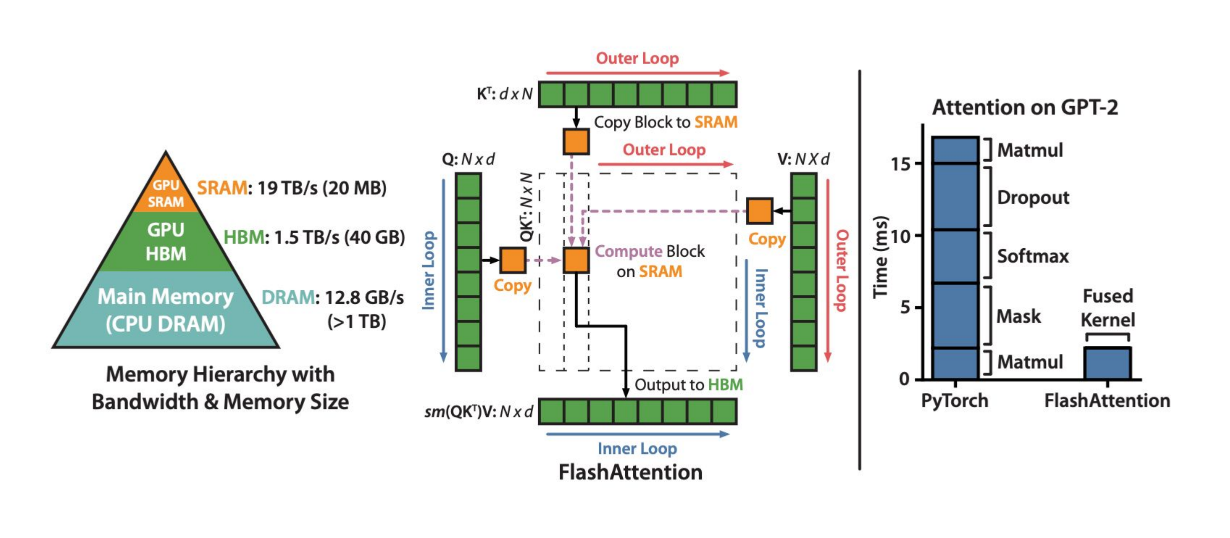
mosaicML MPT-7B Training Data

Data Source	Number of Tokens in Source	Proportion	Effective Number of Tokens	Epochs
mC4 3.1.0 - English (200+ words)	2417.99 B	33%	330 B	0.14
C4 - English - SemDedup 80%	100.42 B	29.9%	299 B	2.98
RedPajama - CommonCrawl	878.45 B	10%	100 B	0.11
The Stack - Selected Languages	463.78 B	10%	100 B	0.22
RedPajama - Wikipedia	4.87 B	4%	40 B	8.21
The Stack - Markdown	107.07 B	3.5%	35 B	0.33
Semantic Scholar ORC	48.95 B	3.3%	33 B	0.68
RedPajama - Books	26.02 B	3%	30 B	1.15
RedPajama - arXiv	28.10 B	1.9%	19 B	0.68
RedPajama - StackExchange	20.54 B	1.4%	14 B	0.68

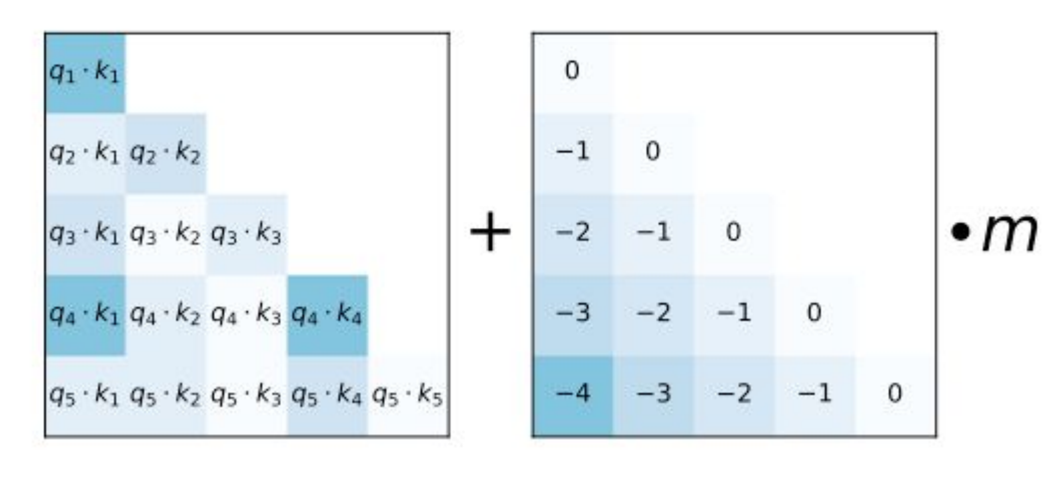
mosaicML Foundation Series

Model Architecture

- Start with the same architecture and model configuration as GPT-3
    - Decoder-only transformer
- Use FlashAttention to reduce memory consumption and increase training throughput
- Use ALiBi instead of positional embeddings, to support long context finetuning and extrapolation
- Rely on small test runs (e.g. MPT-125M, MPT-1B) and use scaling law plots to make decisions



FlashAttention



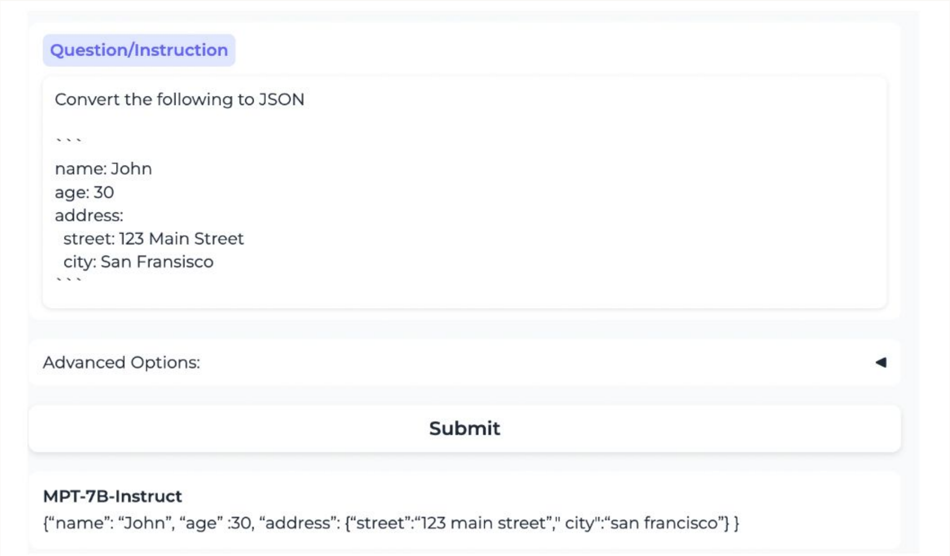
ALiBi attention scores

Finetuning

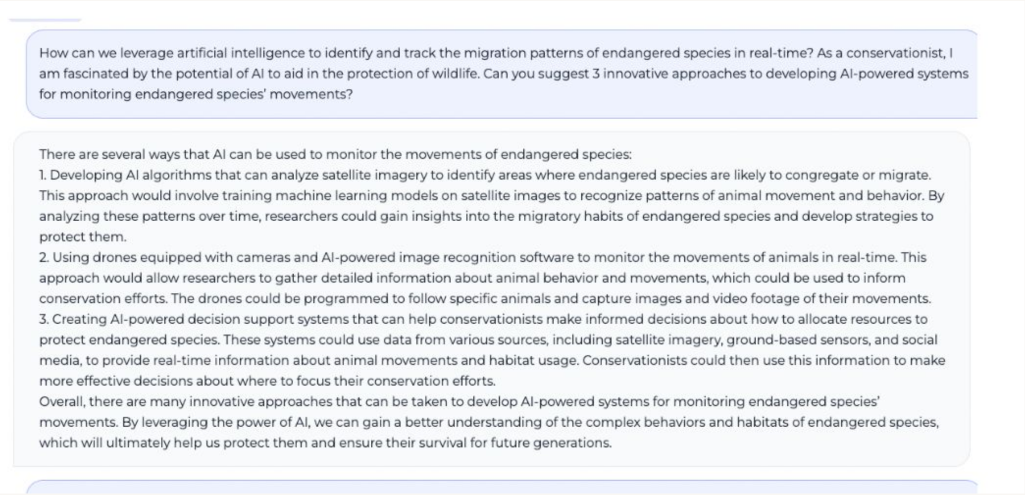
- Take the base model and finetune on a small, curated dataset to make interactions more natural, helpful, safe
    - Short-form instruction following (MPT-7B/30B-Instruct)
    - Multi-turn chat (MPT-7B/30B-Chat)
    - Very long context lengths (MPT-7B-StoryWriter-65K)

- Finetuning runs are much faster and cheaper than pre-training

- Larger base models may require fewer samples to finetune



Sample interaction with MPT-7B-Instruct



Sample interaction with MPT-7B-Chat

MPT-7B Training Details

mosaicML MPT-7B Training Costs

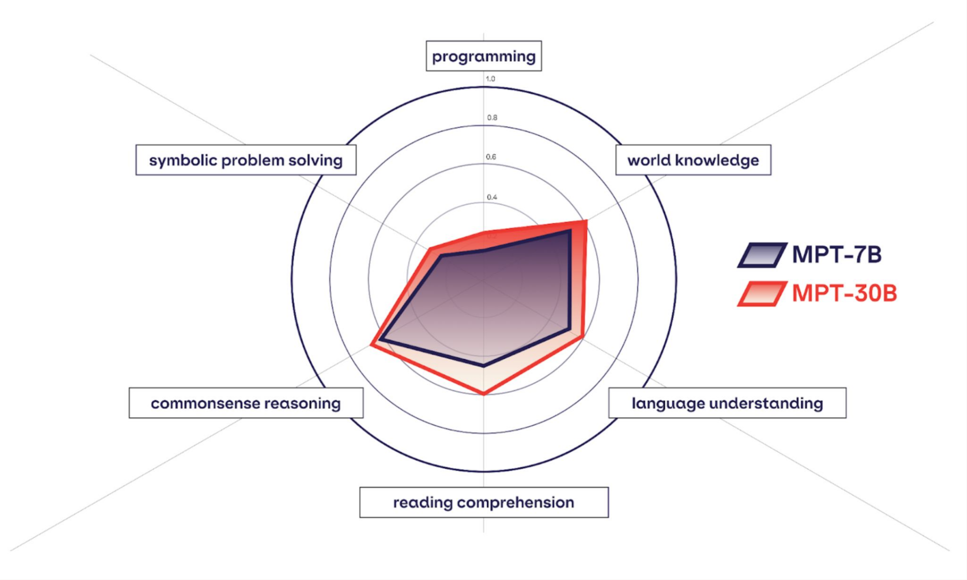
Model	Number of Tokens of Data	Train Batch Size (samples)	Train Context Length	System	Time-to-Train with MosaicML	Cost with MosaicML
MPT-7B	1T	1760	2048	440xA100-40GB	9.5 Days	**$200,640**
MPT-7B-Instruct	9.6M	48	2048	8xA100-40GB	2.3 Hours	**$37**
MPT-7B-Chat	86M	32	2048	8xA100-80GB	8.2 Hours	$164
				+ 32xA100-40GB	6.7 Hours	$429
				Total Combined →	14.9 Hours	**$593**
MPT-7B-StoryWriter-65k+	5B	32	65536	32xA100-80GB	2.2 Days	**$4270**

mosaicML Foundation Series

Eval

- Evaluating and comparing open- and closed-source LLMs is an open problem!
- **Before**: zero-shot and few-shot performance on individual tasks (top)
- **Now**: average performance on large collections of tasks (bottom)
- **Future**: Human rankings of generations, ELO, etc.

Model	LAMBADA (OpenAI)	HellaSwag	PIQA	ARC-Easy	ARC-Challenge	BoolQ	COPA	Winograd	Winogrande	TriviaQA	Jeopardy	MMLU
MPT-7B	0.703	**0.761**	**0.799**	**0.673**	0.394	0.750	**0.813**	**0.878**	**0.683**	0.343	0.308	0.296
LLaMA-7B	**0.738**	0.751	0.792	0.652	**0.411**	**0.767**	0.779	0.807	0.675	**0.443**	**0.334**	**0.302**
StableLM-7B (alpha)	0.533	0.411	0.666	0.435	0.259	0.606	0.672	0.646	0.513	0.049	0.000	0.251
Pythia-7B	0.667	0.636	0.761	0.581	0.325	0.634	0.769	0.786	0.607	0.198	0.022	0.265
Pythia-12B	0.704	0.672	0.768	0.605	0.351	0.675	0.781	0.847	0.627	0.233	0.026	0.253
GPTJ-6B	0.683	0.665	0.762	0.583	0.355	0.648	0.789	0.833	0.641	0.234	0.026	0.261
GPT-NeoX-20B	0.719	0.712	0.780	0.644	0.392	0.691	0.781	0.861	0.665	0.347	0.146	0.269
Cerebras-7B	0.636	0.582	0.744	0.564	0.311	0.625	0.734	0.779	0.603	0.141	0.012	0.259
Cerebras-13B	0.635	0.588	0.740	0.571	0.321	0.611	0.719	0.760	0.602	0.146	0.013	0.258
OPT-7B	0.677	0.676	0.773	0.579	0.329	0.665	0.719	0.840	0.656	0.227	0.020	0.251
OPT-13B	0.692	0.701	0.774	0.586	0.345	0.657	0.805	0.851	0.670	0.282	0.126	0.257



Recap

How much LLMs really cost

Hardware	Precision	Model	Tokens	Time to Train with mosaicML	Cost to Train with mosaicML
512xA100-40GB	AMP_BF16	MPT-30B	1 Trillion	28.3 Days	~ $871,000
512xH100-80GB	AMP_BF16	MPT-30B	1 Trillion	11.6 Days	~ $714,000

Hardware	Precision	Model	Time to Finetune on 1B tokens with MosaicML	Cost to Finetune on 1B tokens with MosaicML
16xA100-40GB	AMP_BF16	MPT-30B	21.8 Hours	$871
16xH100-80GB	AMP_BF16	MPT-30B	8.9 Hours	$714

It’s all about data, scale…and *building* valuable products

Ultimately, the products that will win are the one that create amazing user experiences!

- LLMs are a tool in the toolbox – they will change the interface by which folks interact with our products
  - But ultimately, products must create value for the user
  - Product development should still focus
- LLMs are living data systems, just like your applications – need constant care and improvement
- LLMOps is a real, challenging area with technically deep platform tech