MAX PLANCK INSTITUTE
FOR SOFTWARE SYSTEMS

# ZeRO Mini Project

by Parul Negi & Aseer Ahmad

13 February 2024

Supervisors:

Vedant Nanda
Laurent Bindschaedler
Till Speicher
Krishna P. Gummadi

# Problem Introduction

## GPT-2

GPT-2 is built on the transformer architecture, which was introduced in the paper "Attention is All You Need" by Vaswani et al. It is known for its large scale. The model has 1.5 billion parameters, making it one of the largest language models at the time of its release. The sheer size contributes to its ability to generate coherent and contextually relevant text.OpenAI released GPT-2 with multiple variations, each differing in the number of parameters:
125M: 125 million parameters.
355M: 355 million parameters.
774M: 774 million parameters.
1.5B: 1.5 billion parameters.
It is pre-trained on a diverse range of internet text, allowing it to capture a broad understanding of language and context. The model learns patterns, structures, and information from a wide variety of sources.
GPT-2 and its variations have contributed significantly to the advancement of natural language processing and have paved the way for even larger and more powerful models like GPT-3. Researchers and developers continue to explore ways to improve and responsibly deploy such models for various applications.

## Dataset

databricks-dolly-15k is a corpus of more than 15,000 records generated by thousands of Databricks employees to enable large language models to exhibit the magical interactivity of ChatGPT. Databricks employees were invited to create prompt / response pairs in each of eight different instruction categories, including the seven outlined in the InstructGPT paper, as well as an open-ended free-form category. The contributors were instructed to avoid using information from any source on the web with the exception of Wikipedia (for particular subsets of instruction categories), and explicitly instructed to avoid using generative AI in formulating instructions or responses.

# Relevant Tools

## Hugging Face Transformers

The transformers library is developed by Hugging Face and is widely used for working with pre-trained language models (such as BERT, GPT, etc.) and training custom models. It provides easy access to a large collection of pre-trained models, tokenizers, and utilities for various NLP tasks.

### Torch

PyTorch is an open-source machine learning library developed by Facebook's AI Research lab (FAIR). It is widely used for deep learning and artificial intelligence applications with its key features being Dynamic Computational Graph, Automatic Differentiation , easy of use and pythonic APIs.

### Megatron Deepspeed

DeepSpeed version of NVIDIA's Megatron-LM that adds additional support for several features such as MoE model training, Curriculum Learning, 3D Parallelism, and others.

### Hardware and software

We train using 2 Nvidia A40 GPUs

Docker Container : nvcr.io/nvidia/pytorch:23.12-py3
PyTorch Version 2.2.0a0+81ea7a4
CUDA 12.3 driver version 545.23.08

# Training Methodology

### Dataset Preparation

The dataset has 4 columns namely 'instruction', 'context', 'response' and 'category' . To prepare the dataset we concatenated the following columns in the order :

1. Context
2. Instruction
3. Response

This order of concatenation is only logical to have as any conversation begins with a context followed by an instruction and then finally a response. We would expect our model to infer this behavior only.

Further , GPT2 Tokenizer was used to prepare the data for model input. We also constrain our input sequence length according to available memory on the computing machine.

The dataset preparation is then processed using this shell file : process_databricks_gpt2.sh .

### ZeRO Optimizations and training

We use the torch docker container suggested in the DeepSpeed-megatron documentation along with the latest deepspeed pull to build the project and execute the code.

Further, hyperparameter configurations have been kept the same across all experiments.

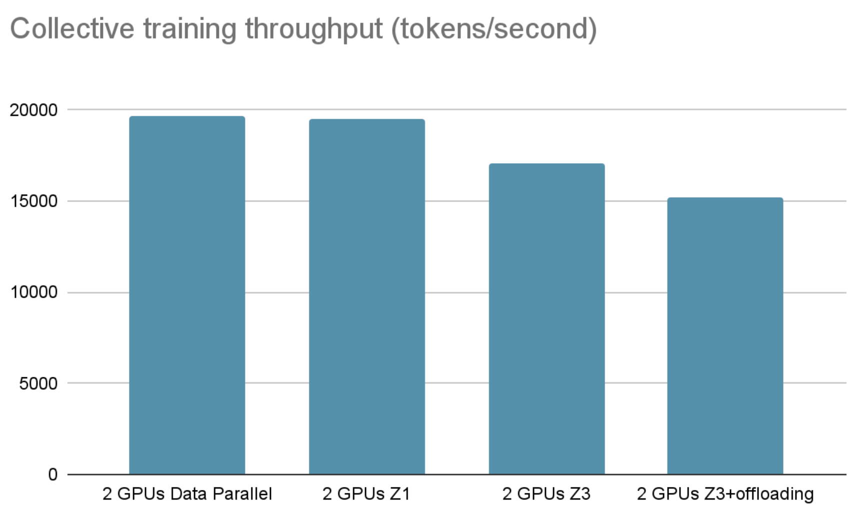
For data parallelism , Zero optimization Stage has been set to 0 which disables it .
For Zero Stage 1 and 3 we set the corresponding stage in the deepspeed configuration . For zero stage 3 and offloading we add an offloading optimizer as CPU along with stage configuration .

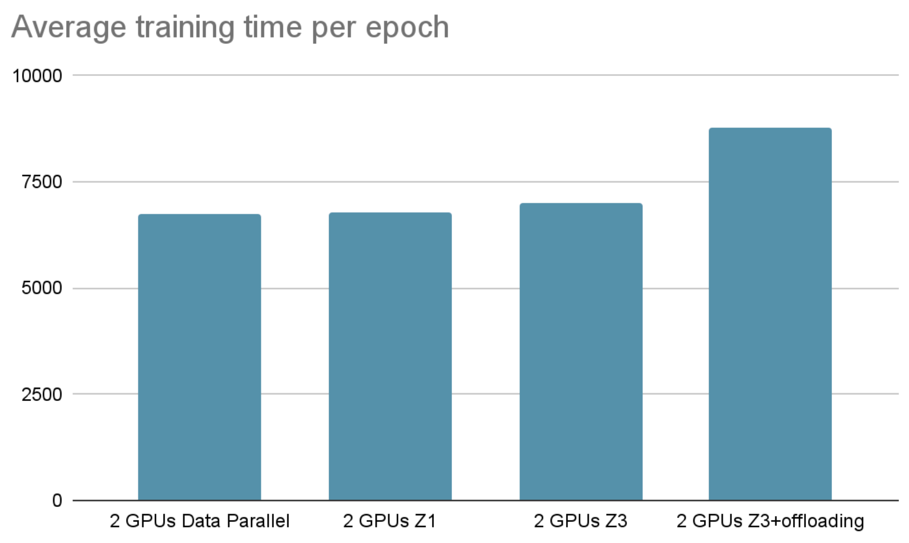
We observe that increasing the mini batch size while keeping the global batch size constant increases the training speed per iteration.

## Analysis

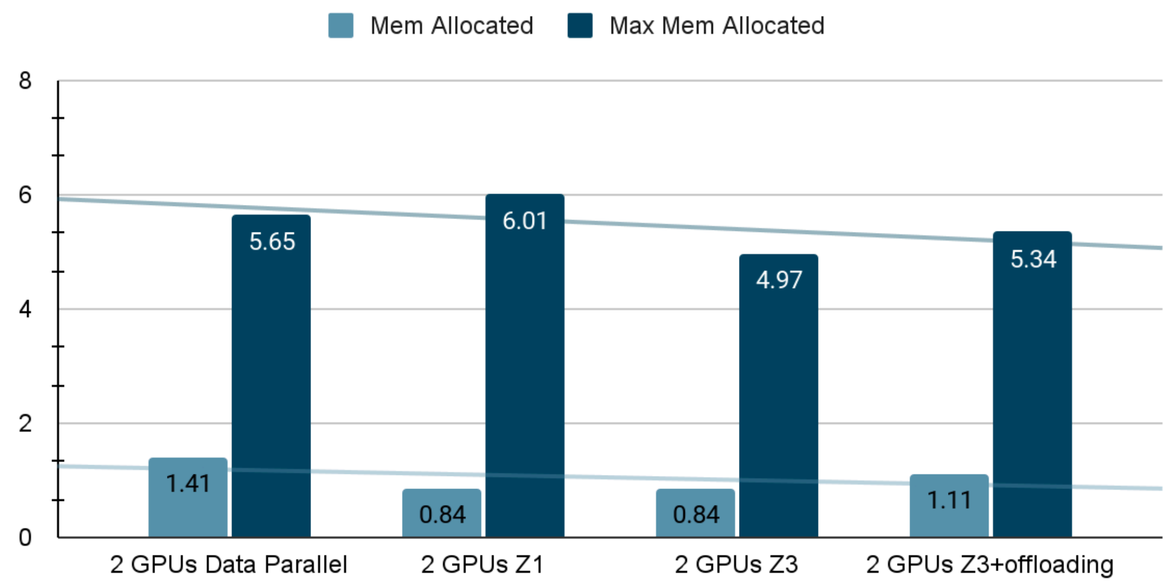
The results have been  collected from the logs generated by DeepSpeed-Megatron . Once the training finishes following regular expressions are used to collect analysis points from the output log and are then summarized :

- `'\(tgs\): (\d+\.\d+)'`
- `'elapsed time per iteration \(ms\): (\d+\.\d+)'`
- `', MemAllocated=(\d+\.\d+)GB'`
- `', MaxMemAllocated=(\d+\.\d+)GB'`
- `'lm loss: (\d+\.\d+[E]\+\d+)'`

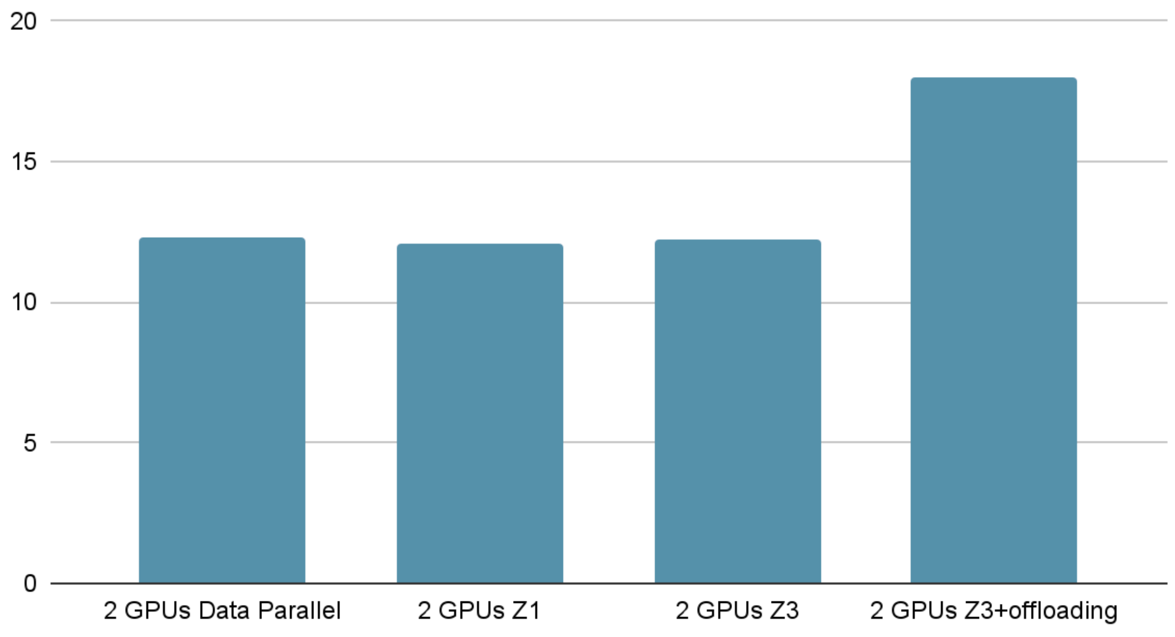

Collective training throughput (tokens/second)
20000
15000
10000
5000
0
2 GPUs Data Parallel
2 GPUs Z1
2 GPUs Z3
2 GPUs Z3+offloading


Average training time per epoch
10000
7500
5000
2500
0
2 GPUs Data Parallel
2 GPUs Z1
2 GPUs Z3
2 GPUs Z3+offloading

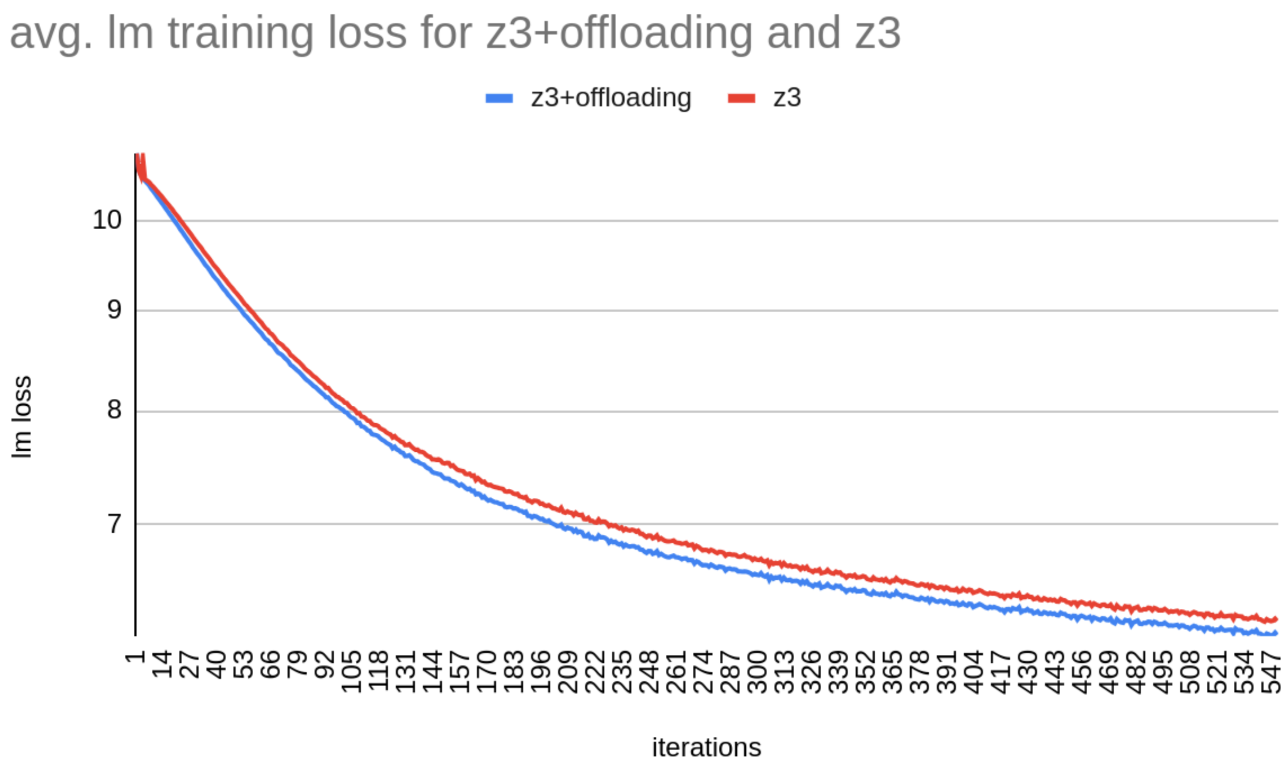
## Avg. GPU memory footprint on each GPU

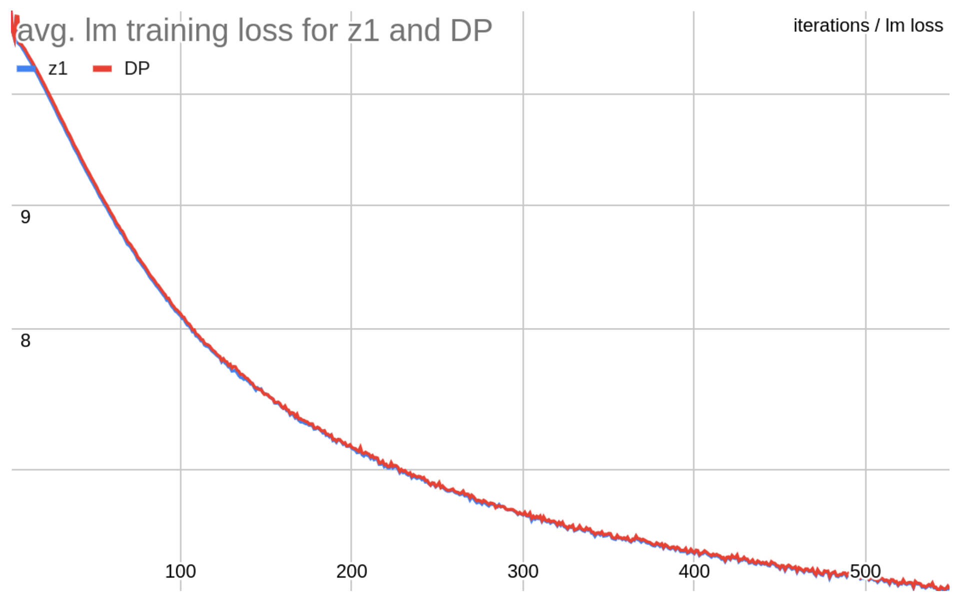


## host memory footprint during training

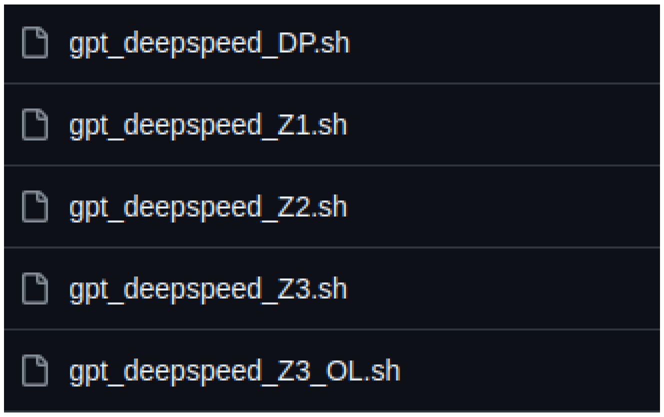


Training loss on the dataset, after 0, 1 epoch of training.





Results of these experiments are stored in a folder labeled `runs/` followed by their corresponding experiment configuration. Similarly to run the code the following files can be run from shell.



gpt_deepspeed_DP.sh

gpt_deepspeed_Z1.sh

gpt_deepspeed_Z2.sh

gpt_deepspeed_Z3.sh

gpt_deepspeed_Z3_OL.sh