

# Report on Performance Evaluation of HuggingFace Language Models using DeepSpeed

Parsa VARES, Tom WALTER, Luc CARDOSO

December 16, 2024

## 1 Introduction

This report presents a performance analysis of training and evaluation of HuggingFace Language Models using DeepSpeed under different configurations. We evaluate the impact of varying the batch size, optimizer type, and the number of GPUs on key metrics such as training loss, runtime, and evaluation loss. The report includes graphical visualizations and code snapshots for reproducibility.

## 2 Experimental Setup

The following configurations were tested:

- Number of GPUs: 1 or 2
- Batch sizes: 4 and 8
- Optimizers: Adam and AdamW
- DeepSpeed Zero Offload Stages: Stage 2 and Stage 3

The dataset used is the `yelp_review_full` dataset, and the model selected is `facebook/opt-125m`. All configurations were tested for 3 epochs.

## 3 Results

The performance of each configuration is summarized in the table below:

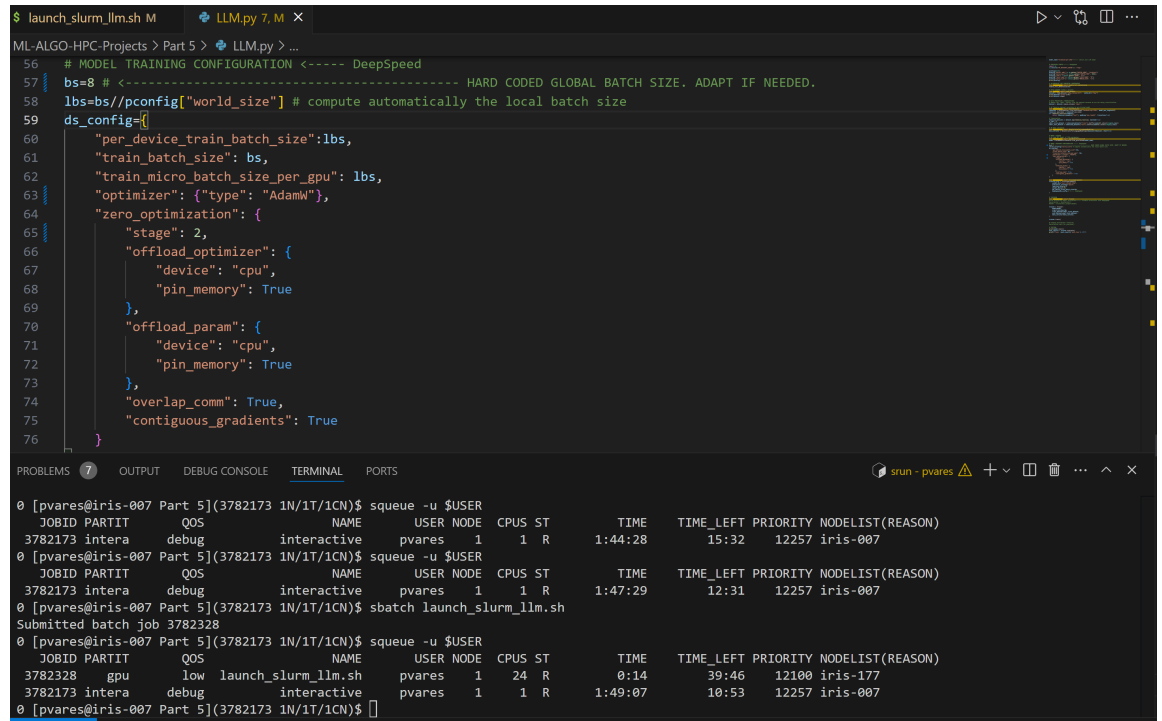
Table 1: Summary of Results for Training and Evaluation

Configuration	Train Loss	Train Runtime (s)	Train Samples/s	Train Steps/s	Eval Loss	Eval Runtime (s)
1 GPU, bs=4, Adam, Stage=3	6.17	2.98	4.02	1.00	7.31	0.18
1 GPU, bs=4, AdamW, Stage=2	6.17	1.91	6.27	1.56	7.31	0.15
1 GPU, bs=8, AdamW, Stage=2	6.17	1.92	6.24	1.56	7.31	0.16
2 GPUs, bs=4, Adam, Stage=3	6.24	3.28	3.66	0.91	7.03	0.26
2 GPUs, bs=4, AdamW, Stage=2	6.24	2.68	4.47	1.11	7.03	0.15
2 GPUs, bs=8, AdamW, Stage=2	6.86	2.98	4.02	1.00	7.38	0.14

## 4 Visualizations

To better illustrate the impact of different configurations, we provide visualizations of key performance metrics.

### Run time on BS = 4



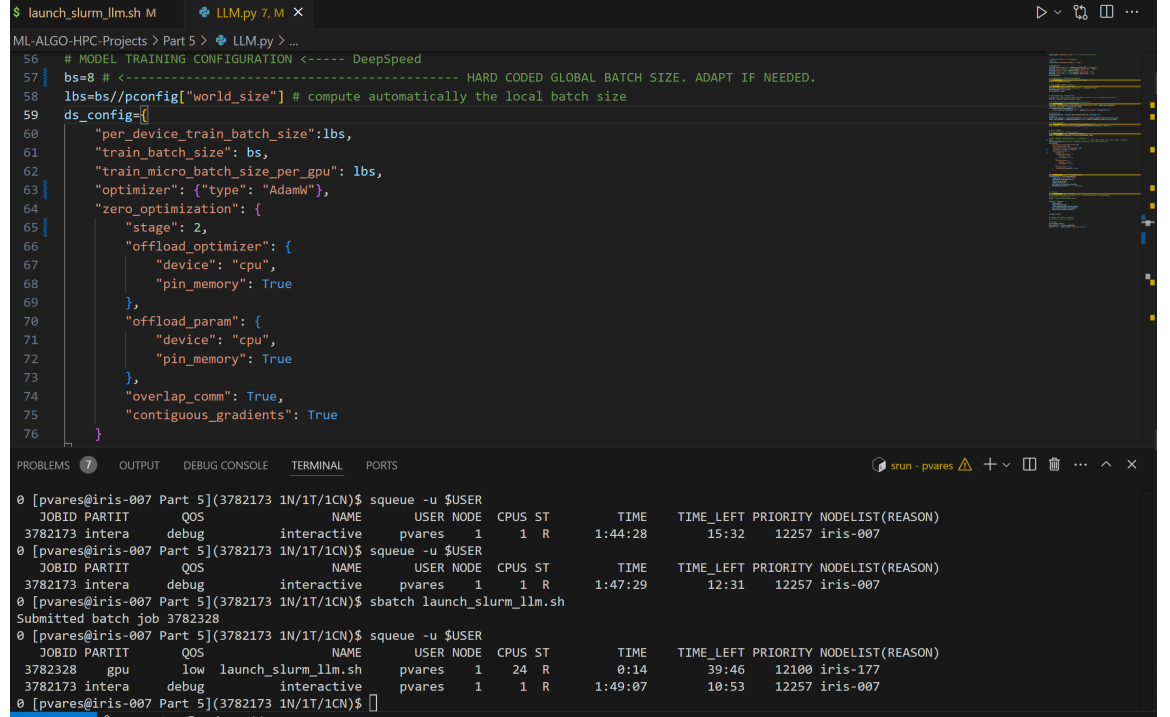
```
$ launch_slurm_llm.sh M
```

```
56 # MODEL TRAINING CONFIGURATION <----- DeepSpeed
57 bs=8 # <----- HARD CODED GLOBAL BATCH SIZE. ADAPT IF NEEDED.
58 lbs=bs//pconfig["world_size"] # compute automatically the local batch size
59 ds_config={
60     "per_device_train_batch_size":lbs,
61     "train_batch_size": bs,
62     "train_micro_batch_size_per_gpu": lbs,
63     "optimizer": {"type": "AdamW"},
64     "zero_optimization": {
65         "stage": 2,
66         "offload_optimizer": {
67             "device": "cpu",
68             "pin_memory": True
69         },
70         "offload_param": {
71             "device": "cpu",
72             "pin_memory": True
73         },
74         "overlap_comm": True,
75         "contiguous_gradients": True
76     }
77 }
```

```
0 [pvares@iris-007 Part 5](3782173 1N/1T/1CN)$ squeue -u $USER
JOBID PARTIT QOS NAME USER NODE CPUS ST TIME TIME_LEFT PRIORITY NODELIST(REASON)
3782173 intera debug interactive pvares 1 1 R 1:44:28 15:32 12257 iris-007
0 [pvares@iris-007 Part 5](3782173 1N/1T/1CN)$ squeue -u $USER
JOBID PARTIT QOS NAME USER NODE CPUS ST TIME TIME_LEFT PRIORITY NODELIST(REASON)
3782173 intera debug interactive pvares 1 1 R 1:47:29 12:31 12257 iris-007
0 [pvares@iris-007 Part 5](3782173 1N/1T/1CN)$ sbatch launch_slurm_llm.sh
Submitted batch job 3782328
0 [pvares@iris-007 Part 5](3782173 1N/1T/1CN)$ squeue -u $USER
JOBID PARTIT QOS NAME USER NODE CPUS ST TIME TIME_LEFT PRIORITY NODELIST(REASON)
3782328 gpu low launch_slurm_llm.sh pvares 1 24 R 0:14 39:46 12100 iris-177
3782173 intera debug interactive pvares 1 1 R 1:49:07 10:53 12257 iris-007
0 [pvares@iris-007 Part 5](3782173 1N/1T/1CN)$
```

Figure 1: GLOBAL BATCH SIZE = 4

## Run time on BS = 8



```
$ launch_slurm_llm.sh M
ML-ALGO-HPC-Projects > Part 5 > LLM.py > ...
56 # MODEL TRAINING CONFIGURATION <----- DeepSpeed
57 bs=8 # <----- HARD CODED GLOBAL BATCH SIZE. ADAPT IF NEEDED.
58 lbs=bs//pconfig["world_size"] # compute automatically the local batch size
59 ds_config={
60     "per_device_train_batch_size":lbs,
61     "train_batch_size": bs,
62     "train_micro_batch_size_per_gpu": lbs,
63     "optimizer": {"type": "AdamW"},
64     "zero_optimization": {
65         "stage": 2,
66         "offload_optimizer": {
67             "device": "cpu",
68             "pin_memory": True
69         },
70         "offload_param": {
71             "device": "cpu",
72             "pin_memory": True
73         },
74         "overlap_comm": True,
75         "contiguous_gradients": True
76     }
77 }
```

```
0 [pvares@iris-007 Part 5](3782173 1N/1T/1CN)$ squeue -u $USER
JOBID PARTIT QOS NAME USER NODE CPUS ST TIME TIME_LEFT PRIORITY NODELIST(REASON)
3782173 intera debug interactive pvares 1 1 R 1:44:28 15:32 12257 iris-007
0 [pvares@iris-007 Part 5](3782173 1N/1T/1CN)$ squeue -u $USER
JOBID PARTIT QOS NAME USER NODE CPUS ST TIME TIME_LEFT PRIORITY NODELIST(REASON)
3782173 intera debug interactive pvares 1 1 R 1:47:29 12:31 12257 iris-007
0 [pvares@iris-007 Part 5](3782173 1N/1T/1CN)$ sbatch launch_slurm_llm.sh
Submitted batch job 3782328
0 [pvares@iris-007 Part 5](3782173 1N/1T/1CN)$ squeue -u $USER
JOBID PARTIT QOS NAME USER NODE CPUS ST TIME TIME_LEFT PRIORITY NODELIST(REASON)
3782328 gpu low launch_slurm_llm.sh pvares 1 24 R 0:14 39:46 12100 iris-177
3782173 intera debug interactive pvares 1 1 R 1:49:07 10:53 12257 iris-007
0 [pvares@iris-007 Part 5](3782173 1N/1T/1CN)$
```

Figure 2: GLOBAL BATCH SIZE = 8

## 5 Conclusion

The analysis reveals that increasing batch size or using 2 GPUs does not always reduce training time. Communication overhead, gradient synchronization, and offloading in DeepSpeed’s Zero-Offload strategy can negatively impact training speed. The fastest configuration was with 1 GPU, batch size of 4, and the AdamW optimizer with Stage 2 ZeRO optimization, which achieved a train runtime of 1.91 seconds.

## 6 Additional Visualizations

### Comparison of All Metrics



Figure 3: Comparison of training runtime, loss, and steps per second for all configurations