Profiling Transformer-Based Model

Patomporn Payoungkhamdee and Wannaphong Phattiyaphaibun

December 17, 2021

1. Introduction

Transformers-based model is the latest state-of-the-art model in machine learning approach of natural language processing. From the observation in Figure 1, increasing the number of GPUs does reduce the training time, but the speed-up factor is fairly low when reaching 4 GPUs. In order to inspect the root cause, we profile the activities of multi-GPUs and a single GPU for fine-tuning WangchanBERTa, a Deep Learning model on a Thai corpus for classification task as well as GPT-2 to benchmark a downstream task. The objective is to test the scalability and runtime of each function by tracing the calling of CUDA operators via NVProf.

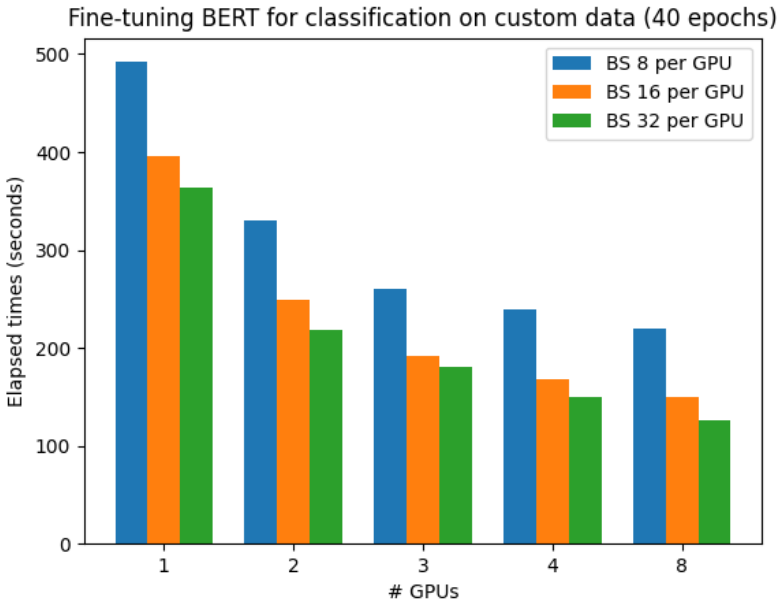


Figure 1. Wall clock time of fine-tuning BERT on various number of graphic cards on DGX

2. Background and Previous Works

This section will walk through the concept of transfer learning that we are using in this work. Secondly, the explanation for the model architecture will be provided. Lastly, we briefly talk about the hardware that we use to benchmark in this work.

2.1. Transfer Learning

Transfer Learning is a method of machine learning. You can use the pre-trained model from the large general dataset to train your dataset or target tasks with a smaller dataset size, which is more efficient than training the model from scratch. [2]

2.2. Transformer, BERT, RoBERTa, and GPT-2

The Transformer is a model architecture of deep learning. It works with stacked self-attention and encoder/decoder stacks. [7]

Bidirectional Encoder Representations (or BERT) from Transformers is sophisticated modified version of encoder part from transformer-based language model. It can be used as a pre-trained model from a large corpus to fine-tune with the specific dataset. [3] For RoBERTa, it is a transformer-based language model with an optimized method from the BERT model. RoBERTa is built on BERT but has different key hyperparameters, implementation, trained with dynamic masking, and removal of BERT's next sentence. [4]

GPT-2 is also a transformer-based language model with 1.5 billion parameters. It trained by large dataset from internet with 8 million web pages. Since it is a bloated model, it can learn with zero-shot setting. [6]

2.3. NVIDIA DGX-1

The testing hardware is NVIDIA DGX-1. It consists of 8×Tesla V100 GPUs, and each of them has 32 GB of attached memory. High level overview of this architecture is illustrated in Figure 2. The characteristic of these multi-GPUs devices is the NVLink. It is the inter-connection to reduce network latency in data sending across the devices. [1]

3. Motivation and Proposed Research

3.1. Motivation

The Transformer-based model has become more and more popular these days. The capability transfer learning is also employed in this model. This technique can save a lot of annotation resources. However, the model is composed of more than a hundred million floating numbers, which can be considered as huge model. It consumes a lot of computational resources. The preliminary observation shows that they are very poor at scalability when adding more GPU

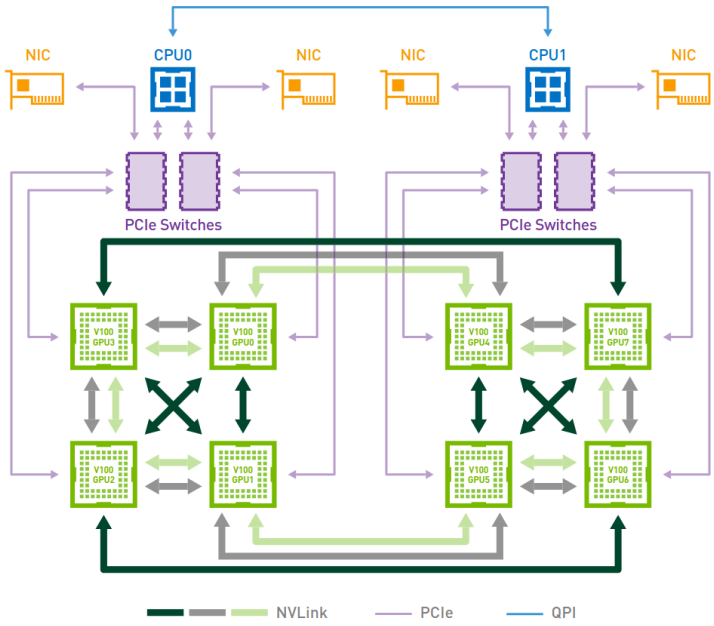


Figure 2. Schematic view of DGX-1. Image taken from [1]

computes units. The hypothesis is that the latency came from communication overhead among the graphical processing units

3.1.1. Our Goal

In order to investigate whether our hypothesis is true or false, we inspect the activities in the running process to find the root cause of the problem.

In summary, we plan to answer the following questions:

1. Why is the speed-up factor when adding more GPU very poor?
2. Does the system introduce more communication latency as the compute unit increases?
3. If there is a communication overhead, which communication interface that the system mostly spend their time in?

3.2. Proposed Idea

We record the running activities from CUDA's interface calling from both built-in and the custom kernel by using NVPROF. This software could track the calling interface on the fly when we train the model, which allows us to trace back to the low-level issue.

3.2.1. Methodology

There are more than a hundred unique interface names from both CUDA built-in and the custom kernel from a developer who developed the technology stack. To simplify the problem, we

divide the instructions into four groups as follows: "MATRIX-MUL" as the matrix multiplicative operator, "MEMORY_MGMT" is the memory management, "CUSTOM_OPS" to be the custom kernel and "OTHER" for the rest of them.

```
def get_ins_group(ops: str) -> str:
    if "gemm" in ops:
        return "matrix-mul"
    elif "CUDA memcpy" in ops or "nccl" in ops.lower() \
        or "copy_device_to_device" in ops.lower() or "CUDA memset" in ops:
        return "memory_mgmt"
    elif "::native" in ops or "vectorized_elementwise" in ops \
        or "_cpp1_ii" in ops or "reduce_kernel" in ops:
        return "custom_ops"
    return "other"
```

We are considering on the fraction of spending time in each group of calling interfaces. If the system and software can run efficiently, the spending time in MEMORY_MGNT would significantly very low comparing to MATRIX-MUL and CUSTOM_OPS and vice versa.

4. Project Timeline

This is our planned timeline for the project. Table 1 shows our planned timeline.

Duration	*Description*
13 October 2021	Submit Project Proposal
14 October - 14 December 2021	Work on Profiling model
15 December 2021	Present Project
17 December 2021	Submit Project Report.

Table 1. Tentative timeline.

5. Profiling Results

We produce the profiling results from our source code, and we release our source code on GitHub.[1]

5.1. RoBERTa

5.1.1. Configurations

The configurations for RoBERTa model are as follows:

[1] https://github.com/calzonelover/Profiling-Transformer-Based-Model

- Batch size: 8
- GPU: 1, 2 and 4
- Model name: Wangchanberta (Thai RoBERTa model) [5]
- Dataset: Wongnai_reviews (Text classification)
- Epoch: 1

5.1.2. Results

The simplest way to observe longest running interface could be done by aggregating each running interface. According to the Table 3, top 10 interfaces in single and double GPUs are all matrix multiplicative operations. On the other hand, most of top 10 interfaces in 4 GPUs are collective communication in the GPU devices which are reduce and broadcast interfaces to transfer the data around the system. This is the first obvious clue to perform further study about how they spend the time on each group of interface to determine the problem from low level aspect.

	Device	Name	Duration
0	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_32x128_tn	135795.181913
1	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x64_nt	131064.973333
2	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x32_nn	129060.355232
3	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_64x32_sliced1x4_nt	123588.278827
4	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x32_sliced1x4_nt	122065.763288
5	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x64_tn	118811.667719
6	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x64_nn	118645.150663
7	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x128_tn	118569.752818
8	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x128_nn	117486.346398
9	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_64x64_tn	57767.068455

(a) 1 GPUs

	Device	Name	Duration
0	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x64_nt	58427.206751
1	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_32x128_tn	58029.424337
2	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x32_nn	55980.616852
3	Tesla V100-SXM2-32GB-LS (1)	volta_sgemm_32x128_tn	55815.426552
4	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_64x32_sliced1x4_nt	55182.212475
5	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x32_sliced1x4_nt	54869.676568
6	Tesla V100-SXM2-32GB-LS (1)	volta_sgemm_128x64_nt	54861.393058
7	Tesla V100-SXM2-32GB-LS (1)	volta_sgemm_128x32_nn	53836.994915
8	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x128_nn	52516.189965
9	Tesla V100-SXM2-32GB-LS (1)	volta_sgemm_64x32_sliced1x4_nt	52288.061033

(b) 2 GPUs

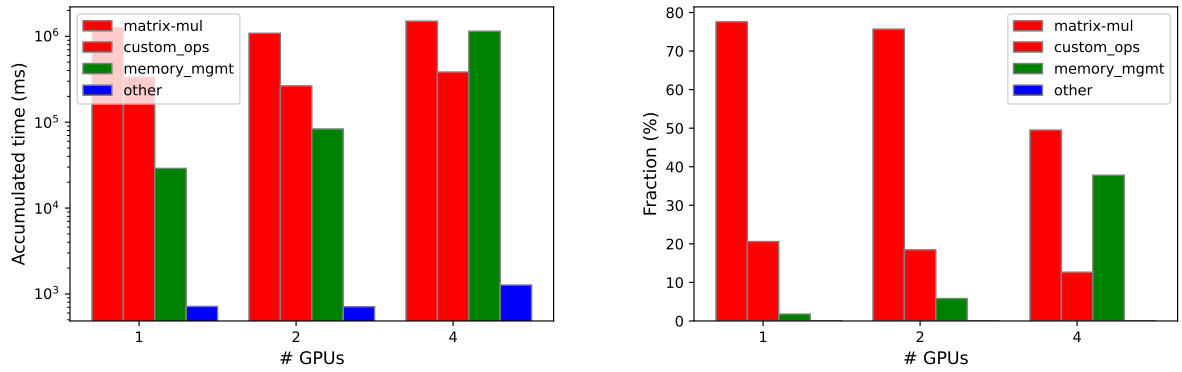
	Device	Name	Duration
0	Tesla V100-SXM2-32GB-LS (0)	ncclReduceRingLLKernel_sum_f32(ncclColl)	157060.957132
1	Tesla V100-SXM2-32GB-LS (2)	ncclReduceRingLLKernel_sum_f32(ncclColl)	156335.110976
2	Tesla V100-SXM2-32GB-LS (3)	ncclReduceRingLLKernel_sum_f32(ncclColl)	155233.676953
3	Tesla V100-SXM2-32GB-LS (1)	ncclReduceRingLLKernel_sum_f32(ncclColl)	152393.073526
4	Tesla V100-SXM2-32GB-LS (1)	volta_sgemm_128x64_tn	127909.728367
5	Tesla V100-SXM2-32GB-LS (2)	ncclBroadcastRingLLKernel_copy_i8(ncclColl)	124997.802063
6	Tesla V100-SXM2-32GB-LS (3)	ncclBroadcastRingLLKernel_copy_i8(ncclColl)	124178.124962
7	Tesla V100-SXM2-32GB-LS (1)	ncclBroadcastRingLLKernel_copy_i8(ncclColl)	123168.228965
8	Tesla V100-SXM2-32GB-LS (3)	volta_sgemm_128x64_tn	122592.641869
9	Tesla V100-SXM2-32GB-LS (0)	ncclBroadcastRingLLKernel_copy_i8(ncclColl)	119301.788980

(c) 4 GPUs

Figure 3. Top 10 longest running interfaces

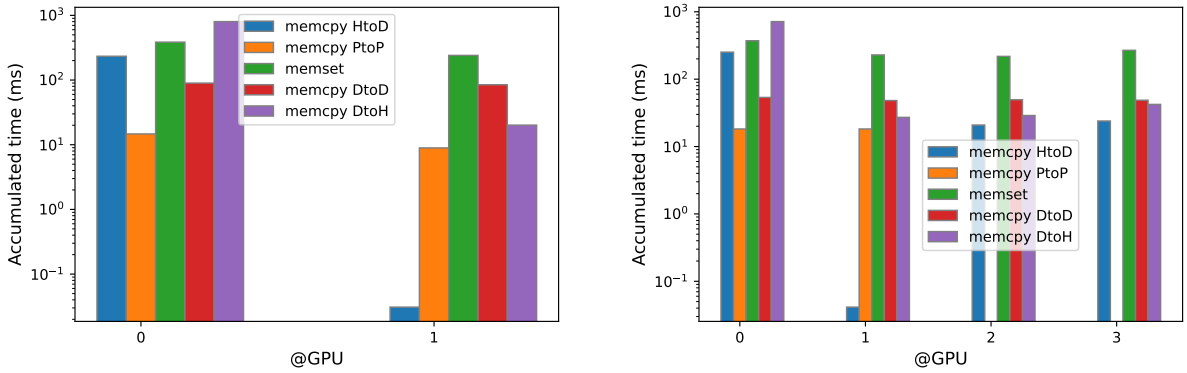
The big picture of the problem can be seen by visualizing the aggregated running time of each operation group as in Figure 4. It can bee seen that increasing the number of GPU will increase

the spending time in memory management operations which is what we suspect in the first place. Moreover, running this model on four GPUs does induce a lot of collective communication across the devices to send the data back and forth. The memory management operation is more than 30% of the overall running time. This is very critical for the scalability. It also implicitly means that the speed up factor after adding more than 4 GPUs might not significant anymore. Hence, we probably under-utilize the resource when training BERT model with more than four GPUs.



(a) Real spending time **(b)** Fraction of time

Figure 4. Spending time of multiple group of operations



(a) 2 GPUs **(b)** 4 GPUs

Figure 5. Aggregated runtime of the built-in memory management operations

Since the memory management operations is likely to be the main factor of the poor speed up factor. Digging into their sub-group operations especially the built-in memory allocation and transfer function could bring us closer to the root cause of the communication overhead. Figure 5 demonstrates the aggregated run-time in each built-in operations. There is an asymmetric behaviour in the memory transfer of host-to-device and device-to-host where the second GPU barely see host-to-device operation for both cases. Moreover, the first GPU tends to volunteer to do this task. Device-to-host operation is more balance with a little asymmetry in the first GPU that they

are significantly does consume more run-time by one order of magnitude.

5.2. GPT-2

5.2.1. Configurations

The configurations for GPT-2 model are as follows:

- Batch size: 4
- GPU: 1, 2 and 4
- Model name: DistilGPT2 (the smallest version of GPT2)
- Dataset: Yelp Dataset (Text Classification)
- Epoch: 1

5.2.2. Results

Similar to BERT model, the top 10 most longest running interfaces in GPT-2 does have the same problem with communication overhead. Considering the Figure 6, The collective communication operators even appears when we train on a couple of GPUs and it getting worse when the number of GPUs reaching to four.

The trivial observations is illustrated in Figure 7, it does have the same problem of data transfer where running on GPUs does spending more than one third of their time. Regarding to the Figure 8, the asymmetric in the host-to-device operation among running devices is more severe. The rest of them barely do host-to-device task. Surprisingly, memset operation is also unbalanced in four GPUs running which is not similar to a two GPUs and all benchmark in BERT.

6. Conclusion

We investigate the kernel/interface calling from the training process. The results show that as the number of GPUs increases, the communication overhead is also introduced to the system. It means that the horizontal scaling for GPU training is highly non-linear. In addition, we also found an asymmetry in spending time on memory management regarding the multi-GPUs regime.

7. References

[1] Nvidia dgx-1 with tesla v100 system architecture. Technical report, NVIDIA, 2017.

[2] J. Brownlee. A gentle introduction to transfer learning for deep learning. *Machine Learning Mastery*, 20, 2017.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

	Device	Name	Duration
0	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x64_tn	1698.524756
1	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x64_nn	1410.022210
2	Tesla V100-SXM2-32GB-LS (0)	void at::native::vectorized_elementwise_kernel...	748.877009
3	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_64x32_sliced1x4_nt	718.032153
4	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x128_nt	570.040032
5	Tesla V100-SXM2-32GB-LS (0)	void at::native::vectorized_elementwise_kernel...	502.679115
6	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_32x128_tn	464.408549
7	Tesla V100-SXM2-32GB-LS (0)	_ZN2at6native27unrolled_elementwise_kernelIZZZ...	459.705535
8	Tesla V100-SXM2-32GB-LS (0)	void at::native::vectorized_elementwise_kernel...	456.498958
9	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_32x128_nn	443.851096

(a) 1 GPUs

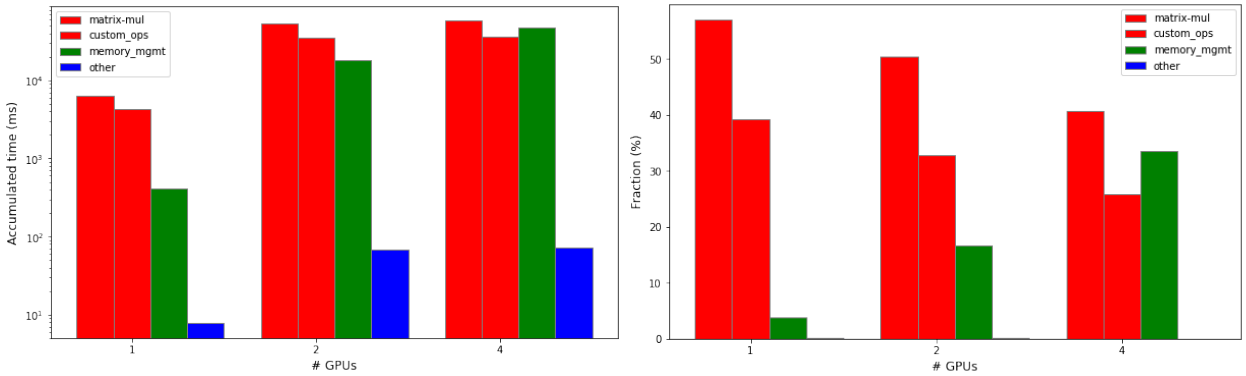
	Device	Name	Duration
0	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x64_tn	7380.638151
1	Tesla V100-SXM2-32GB-LS (1)	volta_sgemm_128x64_tn	7209.306293
2	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x64_nn	6313.397334
3	Tesla V100-SXM2-32GB-LS (1)	volta_sgemm_128x64_nn	6144.265826
4	Tesla V100-SXM2-32GB-LS (1)	ncclBroadcastRingLLKernel_copy_i8(ncclColl)	3700.675143
5	Tesla V100-SXM2-32GB-LS (1)	ncclReduceRingLLKernel_sum_f32(ncclColl)	3646.061010
6	Tesla V100-SXM2-32GB-LS (0)	ncclReduceRingLLKernel_sum_f32(ncclColl)	3635.091898
7	Tesla V100-SXM2-32GB-LS (0)	ncclBroadcastRingLLKernel_copy_i8(ncclColl)	3521.914578
8	Tesla V100-SXM2-32GB-LS (0)	void at::native::vectorized_elementwise_kernel...	3223.492764
9	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_64x32_sliced1x4_nt	3120.157816

(b) 2 GPUs

	Device	Name	Duration
0	Tesla V100-SXM2-32GB-LS (0)	ncclReduceRingLLKernel_sum_f32(ncclColl)	5482.552254
1	Tesla V100-SXM2-32GB-LS (3)	ncclReduceRingLLKernel_sum_f32(ncclColl)	5411.964607
2	Tesla V100-SXM2-32GB-LS (2)	ncclReduceRingLLKernel_sum_f32(ncclColl)	5384.446903
3	Tesla V100-SXM2-32GB-LS (1)	ncclReduceRingLLKernel_sum_f32(ncclColl)	5303.460930
4	Tesla V100-SXM2-32GB-LS (3)	ncclBroadcastRingLLKernel_copy_i8(ncclColl)	4465.193033
5	Tesla V100-SXM2-32GB-LS (2)	ncclBroadcastRingLLKernel_copy_i8(ncclColl)	4452.417370
6	Tesla V100-SXM2-32GB-LS (1)	ncclBroadcastRingLLKernel_copy_i8(ncclColl)	4372.958164
7	Tesla V100-SXM2-32GB-LS (0)	ncclBroadcastRingLLKernel_copy_i8(ncclColl)	4208.870268
8	Tesla V100-SXM2-32GB-LS (1)	volta_sgemm_128x64_tn	3991.212500
9	Tesla V100-SXM2-32GB-LS (0)	volta_sgemm_128x64_tn	3975.629278

(c) 4 GPUs

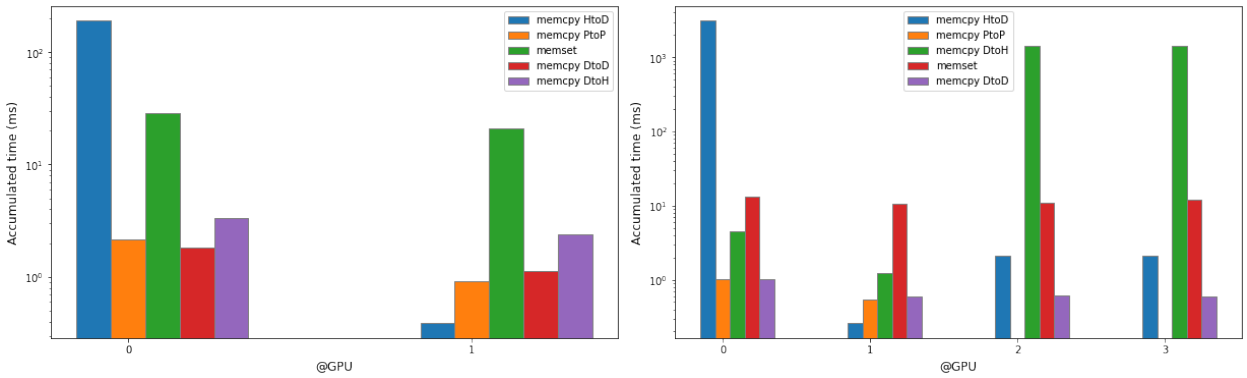
Figure 6. Top 10 longest running interfaces



(a) Real spending time

(b) Fraction of time

Figure 7. Spending time of multiple group of operations



(a) 2 GPUs **(b)** 4 GPUs

Figure 8. Aggregated runtime of the built-in memory management operations

[5] L. Lowphansirikul, C. Polpanumas, N. Jantrakulchai, and S. Nutanong. Wangchanberta: Pretraining transformer-based thai language models. *CoRR*, abs/2101.09635, 2021.

[6] A. Radford, J. Wu, D. Amodei, D. Amodei, J. Clark, M. Brundage, and I. Sutskever. Better language models and their implications. *OpenAI Blog https://openai.com/blog/better-language-models*, 1:2, 2019.

[7] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

8. Appendix

Hardware

- Nvidia DGX-1
- GPU: Tesla V100 x 8
- CPU: Intel(R) Xeon(R) CPU E5-2698 v4 @ 2.20GHz
- RAM: 503 GB

Software

- nvprof 10.2.89
- python 3.6
- CUDA NVIDIA-SMI 450.119.04
- Driver Version: 450.119.04
- CUDA Version: 11.0
- Running on (Docker) Container
- Docker image: nvcr.io/nvidia/pytorch:19.11-py3