DYNAMIC STASHING QUANTIZATION FOR EFFICIENT TRANSFORMER TRAINING

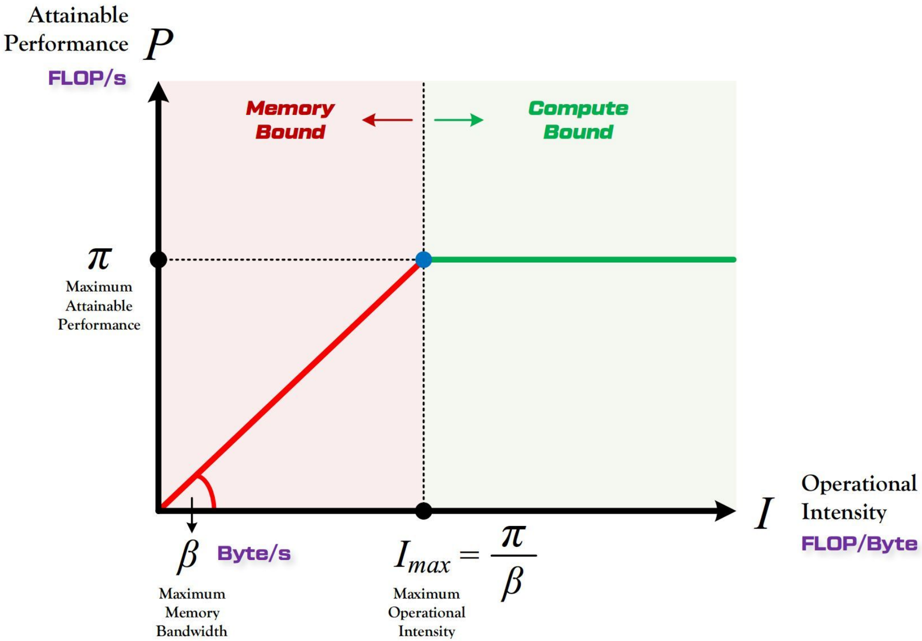
Guo Yang, Daniel Lo, Robert Mullins, Yiren Zhao

Content List

1. Motivation: Roofline Model, Quantization and Stashing
2. Introduction: Transformer Structure, Data Type
3. Methodology: BFP Quantization, Stashing (static vs. dynamic)
4. Results
5. Limitation
6. Summary

1. Motivation: Roofline Model

- Roofline Model [1]:
  - Operational Intensity: performance of computing core over bandwidth of memory: OI := FLOPs per sec/ Memory Access per sec
  - For a specific hardware, the optimal working operational intensity is I_{max}.



[1] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. Communications of the ACM, 52(4):65–76, 2009.

1. Motivation: LLM Training is Memory-Bound

- We ran a similar analysis to Ivanov et al. [1]:

OPT-1.3B Model:

  - 47.93% latency: Attention layers. Memory-bound. (the matrix multiplications only takes around 66% of the latency in a single attention module.)
  - 32.20% latency: FC layer. Compute-bound.
  - 19.87% latency: Activation & Norm layer. Memory-bound.

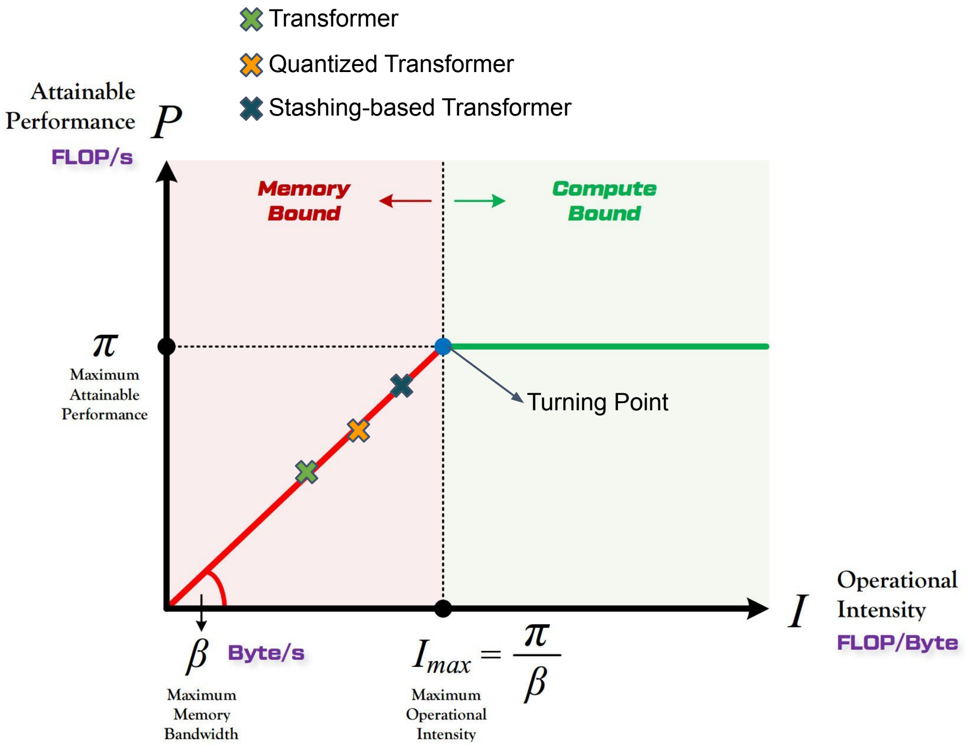
[1] Andrei Ivanov, Nikoli Dryden, Tal Ben-Nun, Shigang Li and Torsten Hoefler. 2020. Data movement is all you need: A case study on optimizing Transformers. In Proceedings of the 4 th MLSys Conference, San Jose, CA, USA, 2021.

1. Motivation: Quantization & Stashing

- Transformer model falls at memory-bound area when trained on modern GPUs.

- Quantization reduces the size of data transmitted between memory and computing core during training.

- Dynamic Stashing Quantization: dynamically quantize the intermediate results between forward and backward passes for a significant reduction of the DRAM traffic



2. Introduction: Transformer Structure

- A Transformer is built up with N stacks of encoders and decoders (Figure 1) [1].
- Encoders and decoders are both composed of three distinct layers:
  - Feed-Forward Network Layer
  - Multi-Head Attention Layer
  - Add & Norm Layer
- FFN Layer and Self-Attention layer consumes most computational resources.

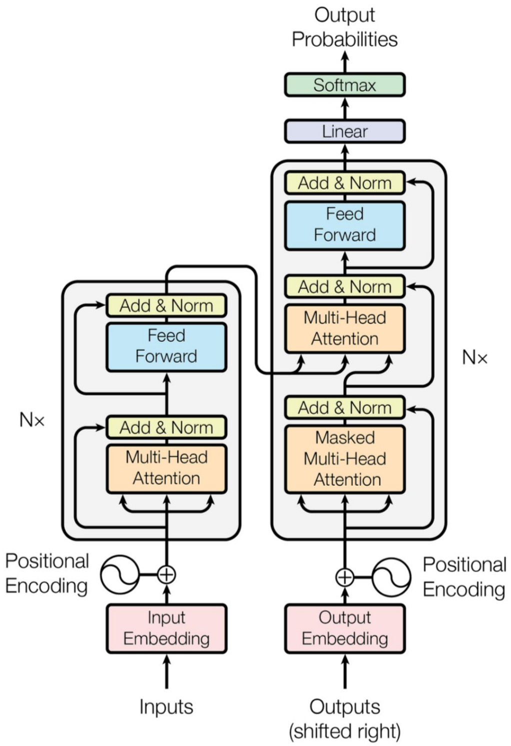


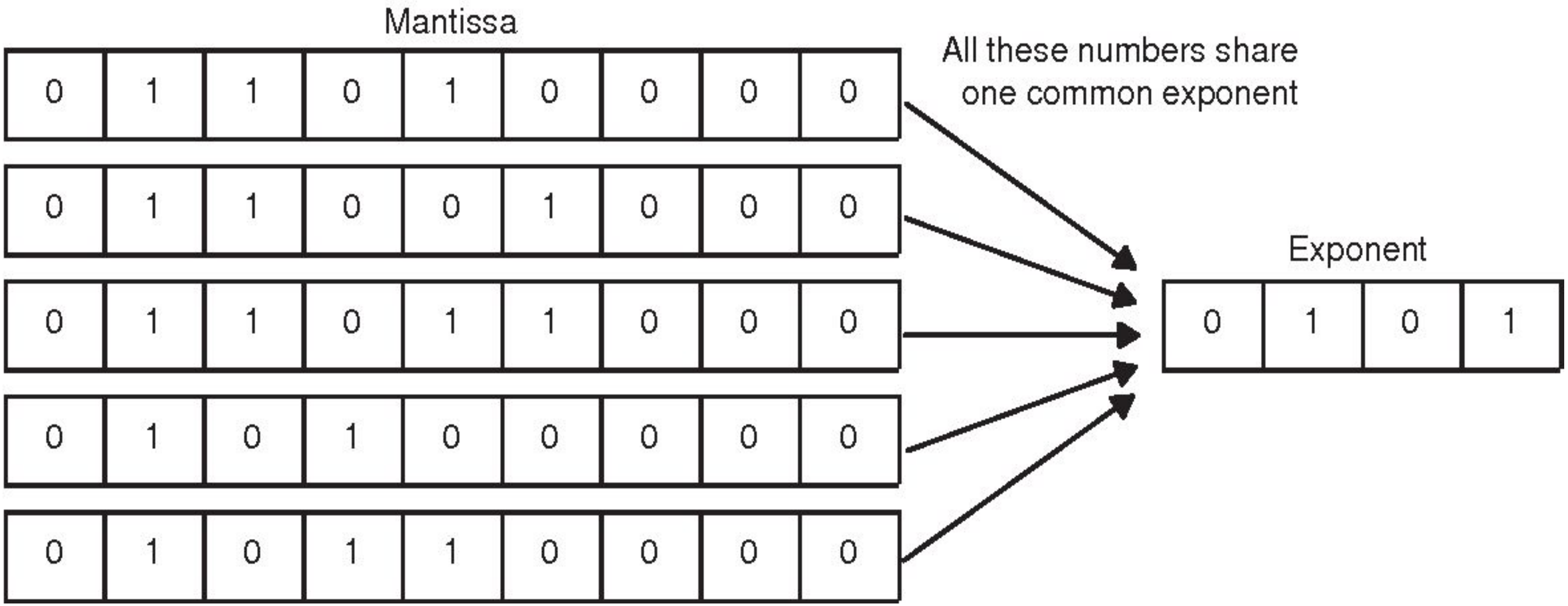
Figure 1: The Transformer - model architecture.

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

2. Introduction: Data Type

- Data Type: INT, FP16, FP32, BFloat16, Block FP (BFP)
- Fixed-Point Quantization: Map Floating-Point values to Fixed-Point values
- BFP Quantization: Quantize the Floating-Point values in groups based on the largest value in every group (Bounding-Box)

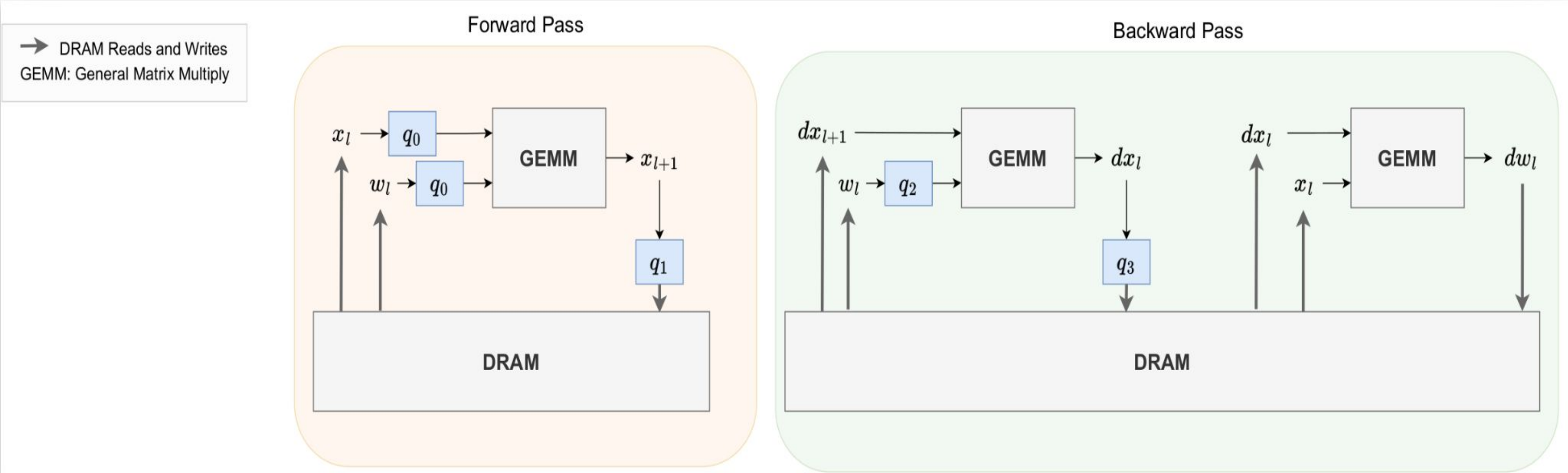
3. Methodology: BFP Quantization (Figure [1])



[1] Chhabra, A., & Iyer, R. (1999). A Block Floating Point Implementation on the TMS 320 C 54 x DSP.

3. Methodology: Stashing

- Stashing:



- Static vs. dynamic

4. Results

Dataset and Model	Method	Precision Setup	Acc / BLEU (Δ)	Arith Ops ($\downarrow$)	DRAM R/W ($\downarrow$)
IWSLT2017 DE-EN Transformer (6-layer)	Floating-point	[32, 32, 32, 32]	35.22	-	-
	Fixed-point	[32, 32, 32, 32]	34.47 (-0.75)	1.00×	1.00×
	Fixed-point	[16, 16, 16, 16]	32.59 (-2.63)	0.25×	0.50×
	Block FP	[32, 32, 32, 32]	34.56 (-0.66)	0.56×	1.13×
	Block FP	[16, 16, 16, 16]	34.30 (-0.92)	0.18×	0.63×
	Stashing (Fixed)	[16, 4, 4, 16]	25.50 (-9.72)	0.13×	0.31×
	Stashing (BFP)	[16, 4, 4, 16]	34.78 (-0.44)	0.10×	0.45×
	DSQ (BFP)	–	34.81 (-0.41)	0.012×	0.20×
GLUE MNLI RoBERTa-base	Floating-point	[32, 32, 32, 32]	87.6	-	-
	Fixed-point	[32, 32, 32, 32]	87.9 ($+0.3$)	1.00×	1.00×
	Fixed-point	[16, 16, 16, 16]	87.9 ($+0.3$)	0.25×	0.50×
	Block FP	[32, 32, 32, 32]	87.8 ($+0.2$)	0.56×	1.13×
	Block FP	[16, 16, 16, 16]	87.8 ($+0.2$)	0.18×	0.63×
	Stashing (Fixed)	[16, 4, 4, 16]	82.8 (-4.8)	0.13×	0.32×
	Stashing (BFP)	[16, 4, 4, 16]	87.8 ($+0.2$)	0.10×	0.45×
	DSQ (BFP)	–	87.8 ($+0.2$)	0.043×	0.26×
GLUE QNLI RoBERTa-base	Floating-point	[32, 32, 32, 32]	92.8	-	-
	Fixed-point	[32, 32, 32, 32]	92.6 (-0.2)	1.00×	1.00×
	Fixed-point	[16, 16, 16, 16]	92.6 (-0.2)	0.25×	0.50×
	Block FP	[32, 32, 32, 32]	92.7 (-0.1)	0.56×	1.13×
	Block FP	[16, 16, 16, 16]	92.5 (-0.3)	0.18×	0.63×
	Stashing (Fixed)	[16, 4, 4, 16]	89.5 (-3.3)	0.13×	0.32×
	Stashing (BFP)	[16, 4, 4, 16]	92.6 (-0.2)	0.10×	0.45×
	DSQ (BFP)	–	92.7 (-0.1)	0.043×	0.26×

5. Limitation

- Scheduling precision: We follow a setup similar to that proposed by Honig et al. [1]. We introduce the parameter N, and found that setting it to N=5 is sufficient for all test scenarios.

- Explore other schedules…

UNIVERSITY OF CAMBRIDGE

[1] Robert Hönig, Yiren Zhao, and Robert Mullins. 2022. Dadaquant: Doubly-adaptive quantization for communication-efficient federated learning. In International Conference on Machine Learning, pages 8852–8866. PMLR

6. Summary

- We propose Dynamic Stashing Quantization (DSQ) for LLM training
- DSQ reduces DRAM traffic by quantizing intermediate results between the forward and backward passes generated during training.
- DSQ keeps most of the model accuracy.
- We demonstrate the effectiveness of DSQ by showing how it can reduce both the computation cost and DRAM bandwidth requirement on machine translation and LLM fine-tuning tasks.

Thank you for listening!

Reference:

[1] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. Communications of the ACM, 52(4):65–76, 2009.

[2] Andrei Ivanov, Nikoli Dryden, Tal Ben-Nun, Shigang Li and Torsten Hoefler. 2020. Data movement is all you need: A case study on optimizing Transformers. In Proceedings of the 4 th MLSys Conference, San Jose, CA, USA, 2021.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[4] Chhabra, A., & Iyer, R. (1999). A Block Floating Point Implementation on the TMS 320 C 54 x DSP.

[5] Robert Hönig, Yiren Zhao, and Robert Mullins. 2022. Dadaquant: Doubly-adaptive quantization for communication-efficient federated learning. In International Conference on Machine Learning, pages 8852–8866. PMLR