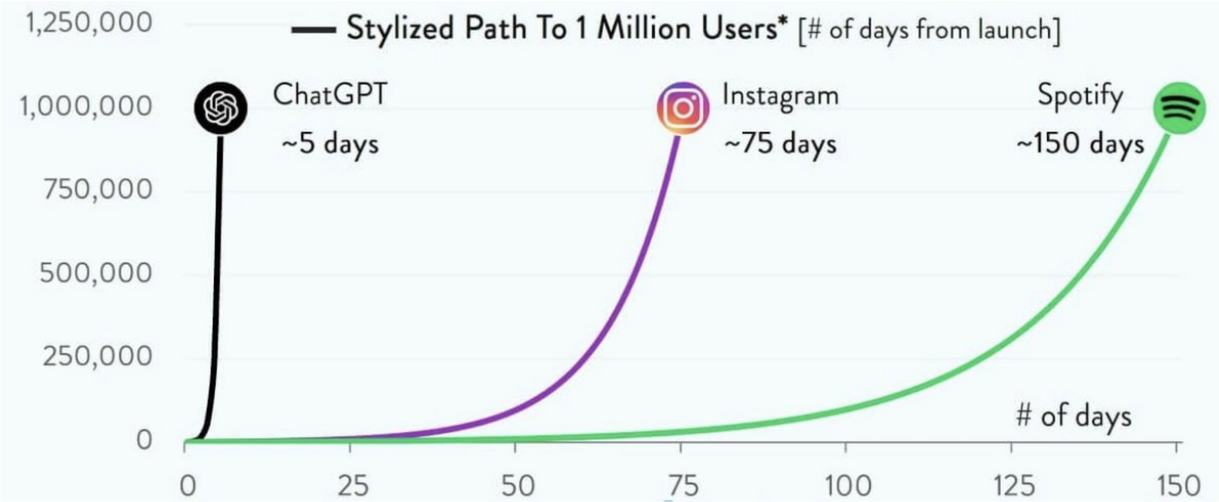
Atom: Low-Bit Quantization for Efficient and Accurate LLM Serving

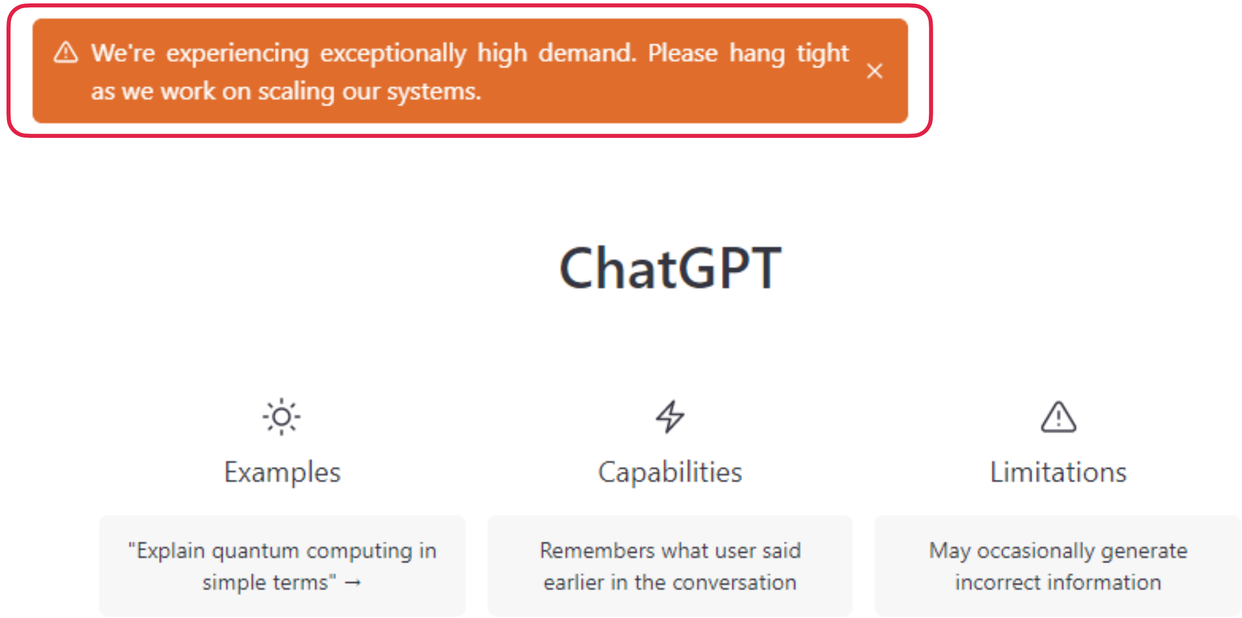
Yilong Zhao, Chien-Yu Lin, Kan Zhu, Zhao Ye, Lequn Chen, Size Zheng,
Luis Ceze, Arvind Krishnamurthy, Tianqi Chen, Baris Kasikci

Presenter: Yilong Zhao 11.30.2023

LLM Serving is at capacity…



GPT gains more popularity.

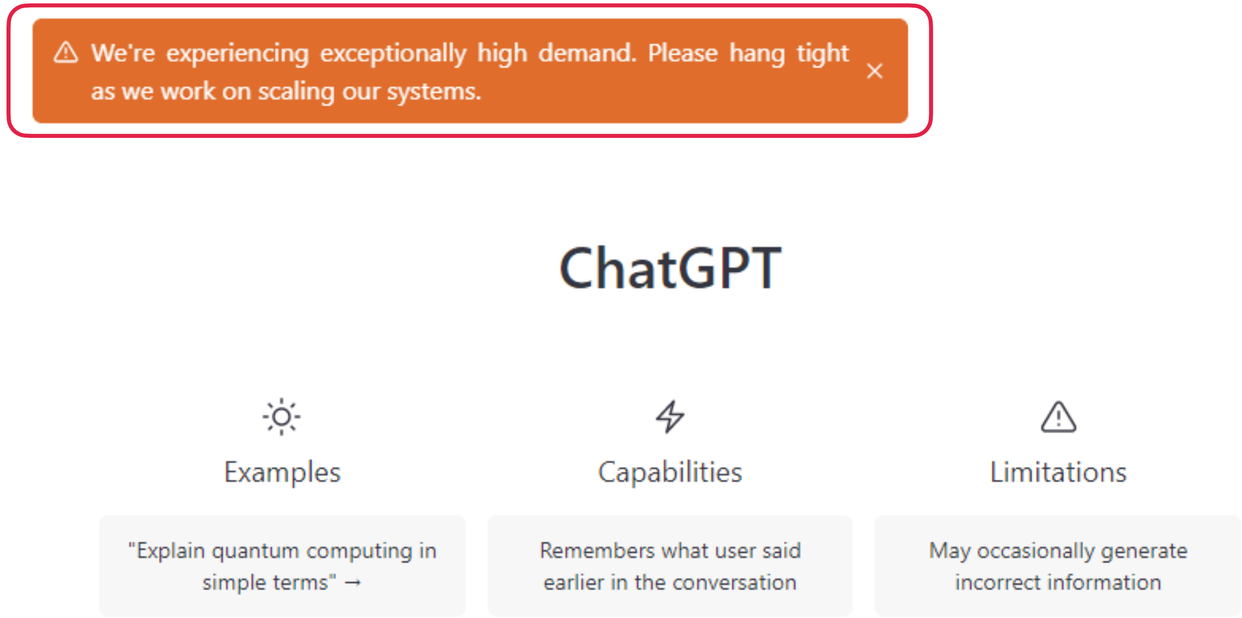


However, serving such high demand is difficult.

[1] https://twitter.com/Contrary_Res/status/1610752946412847104

LLM Serving is at capacity…

Okay, let's buy more machines.



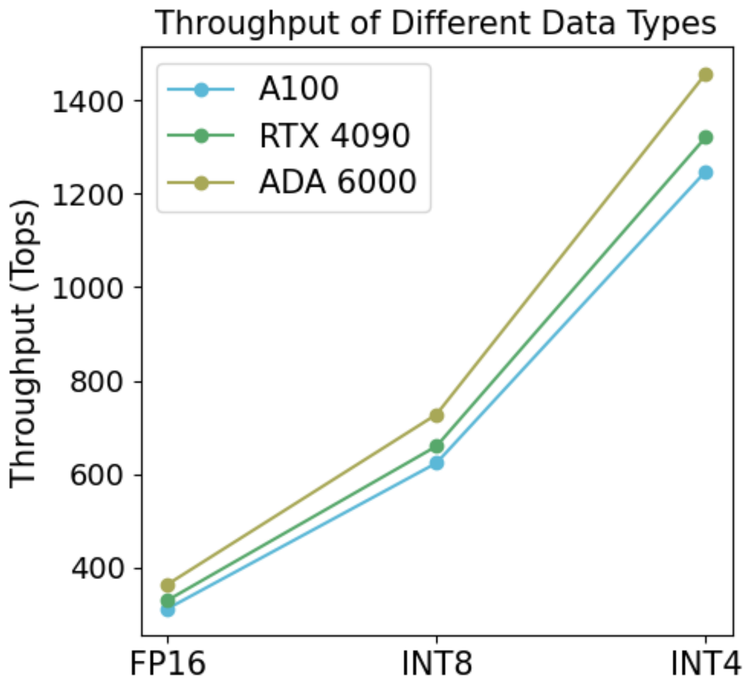
However, serving such high demand is difficult.

[1] All memes are generated by ChatGPT.

LLM Serving is at capacity…

~~Okay, let's buy more machines.~~

Hold on, do we fully utilize existing GPUs?



Efficient support for low-bit calculation.

[1] All memes are generated by ChatGPT.

LLM Serving is at capacity…

~~Okay, let's buy more machines.~~

Well, why not use quantization?

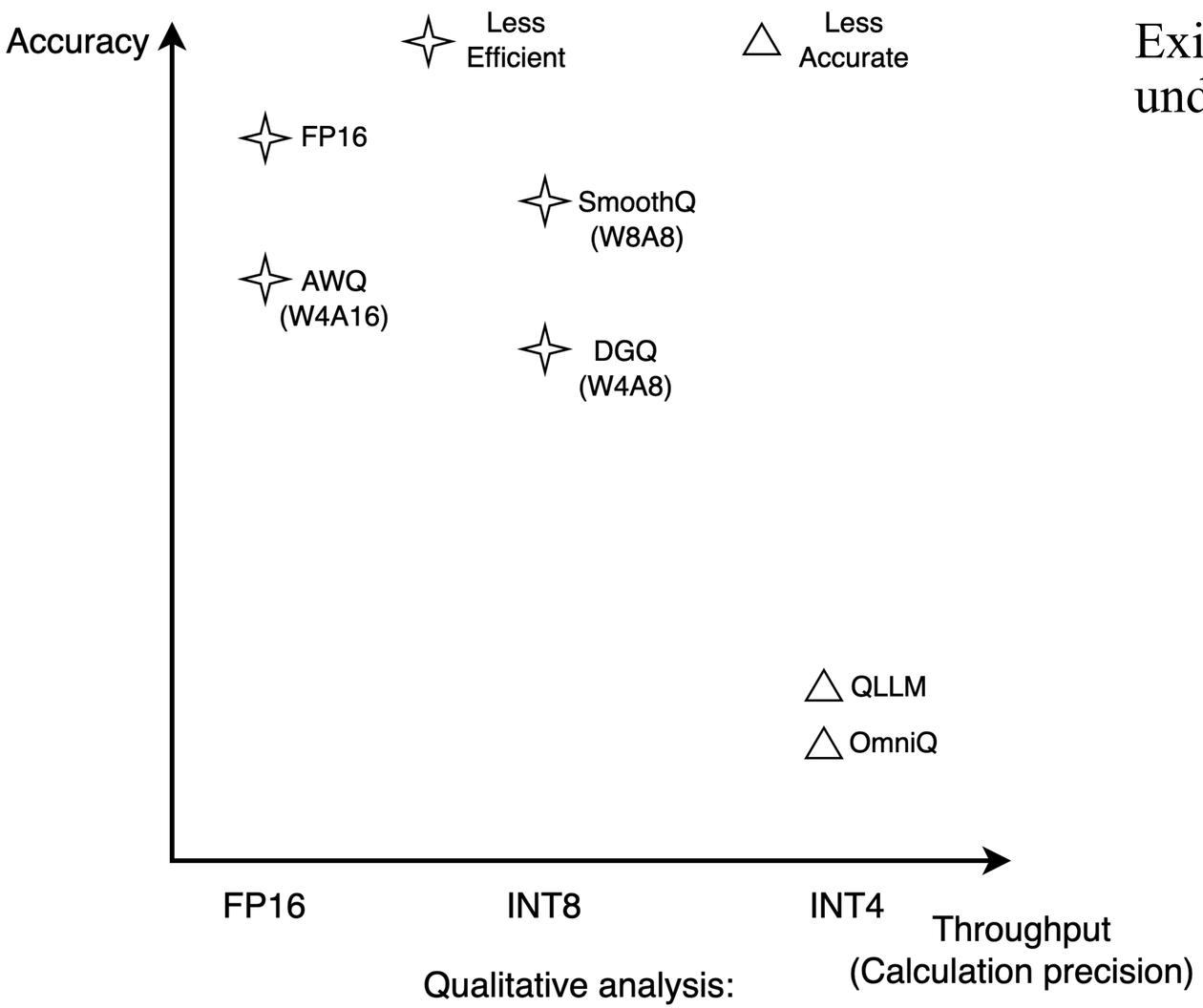
RTX4090 Specifications:

	FP16 🤒	INT4 😄	
Tops	165.2	1321.2	8x
Memory Elements/s	504 G	2016 G	4x

RTX4090 Memory Bandwidth: 1008 GB/s

[1] All memes are generated by ChatGPT.

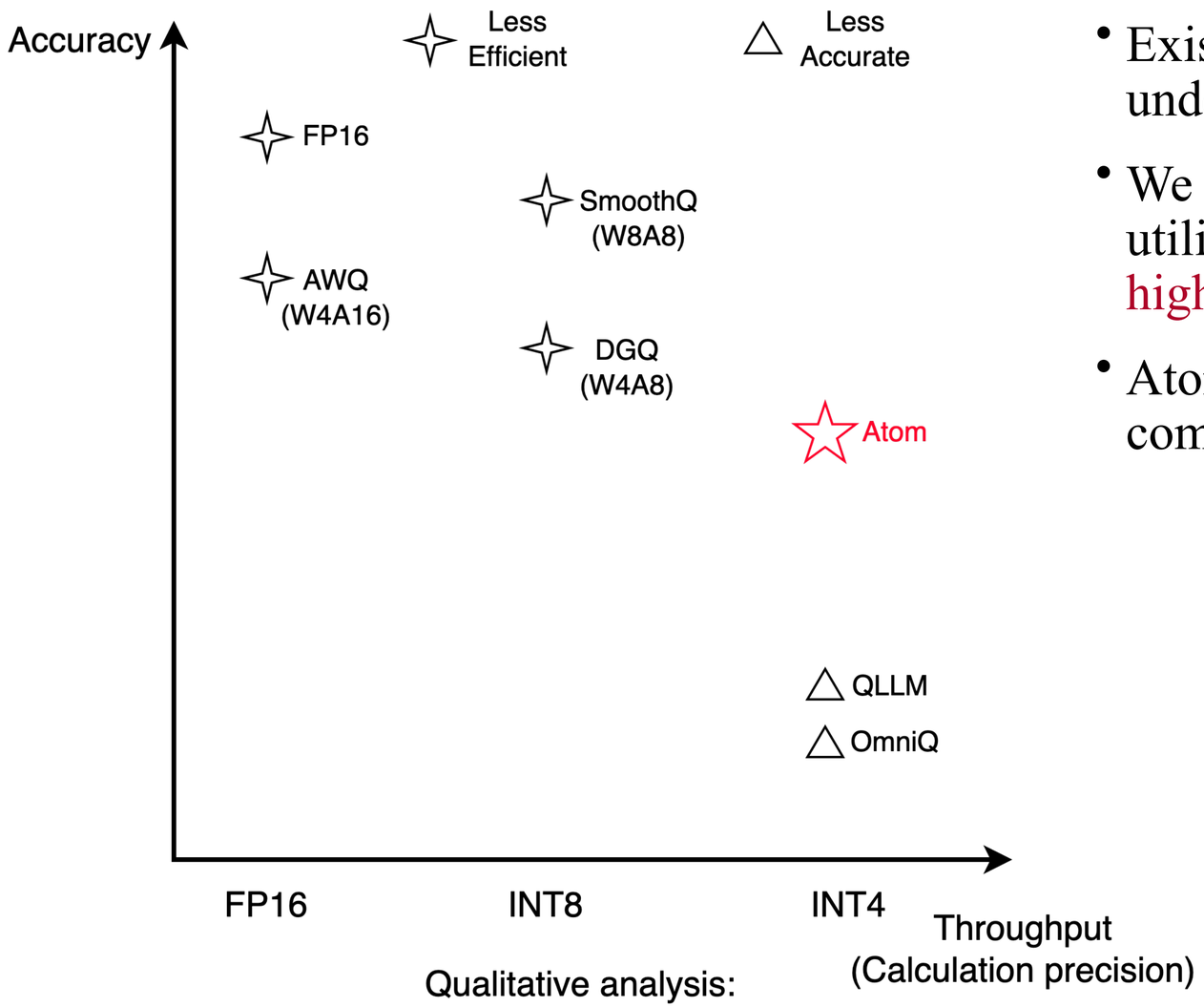
Motivations: Why Atom is needed?



Qualitative analysis: accuracy v.s. throughput

Existing mechanism either not fully utilizes the underlying hardware, or loss too much accuracy.

Motivations: Why Atom is needed?



Qualitative analysis: accuracy v.s. throughput

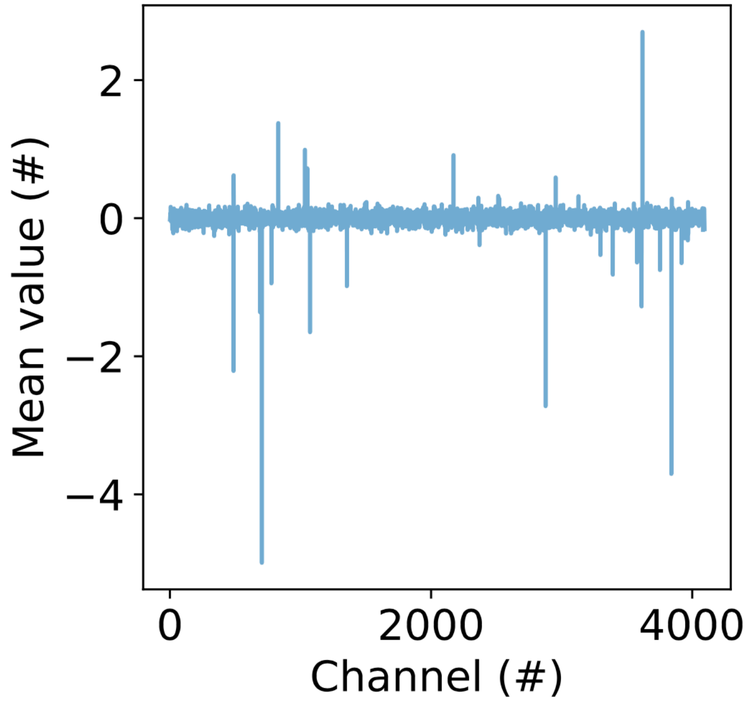
- Existing mechanism either not fully utilizes the underlying hardware, or loss too much accuracy.
- We introduce Atom, a quantization mechanism utilizing INT4 hardware support while maintaining high accuracy.
- Atom boosts end-to-end throughput by 7.7x compared with FP16, 2.5x compared with SmoothQ

Outlines

- Algorithm design of Atom
- Implementation: CUDA Kernels
- Evaluation & Performance Analysis
- Discussion & Future works

Algorithm design: How to keep accuracy with 4-bit precision?

- Several studies[1] show the outlier phenomena in activation tensors.
- Certain amount of channels (i.e., columns) have much larger value than others, typically 10x.

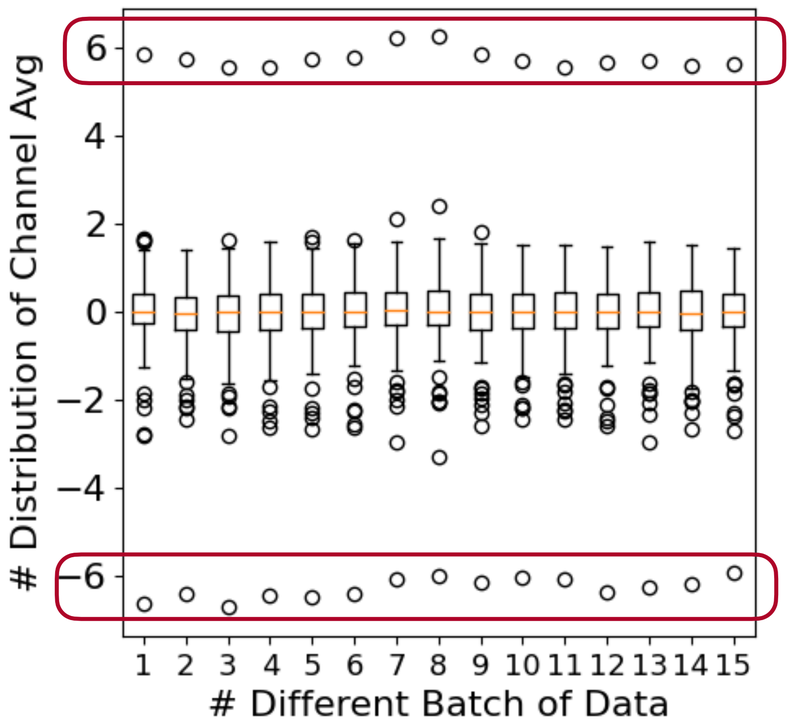


Sample Data of Single Batch

[1] LLM.int8(), SmoothQuant, AWQ, OmniQuant, QLLM…

Outliers in Activations

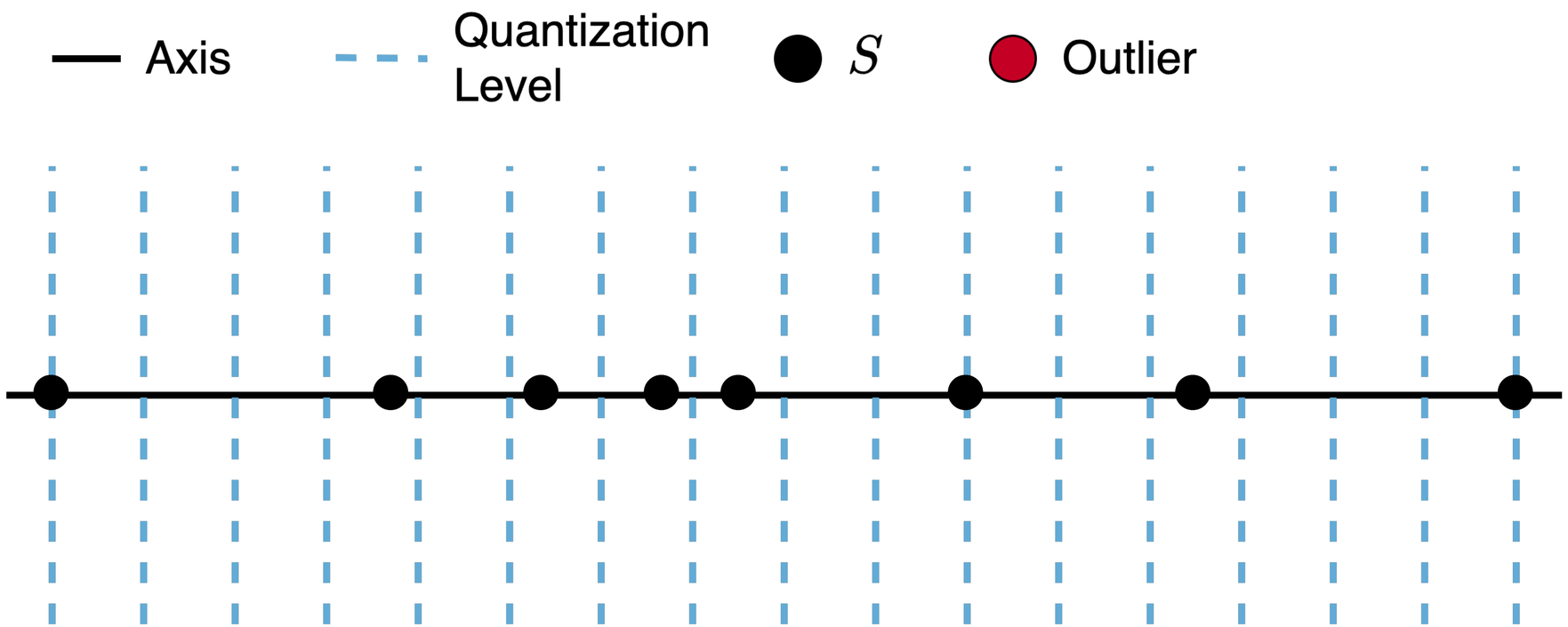
- Several studies[1] show the outlier phenomena in activation tensors.
- Certain amount of channels have much larger value than others, typically 10x.
- This phenomena is consistent with different batches of input data.



[1] LLM.int8(), SmoothQuant, AWQ, OmniQuant,…

Outliers: 1. It will squeeze the accuracy of other values.

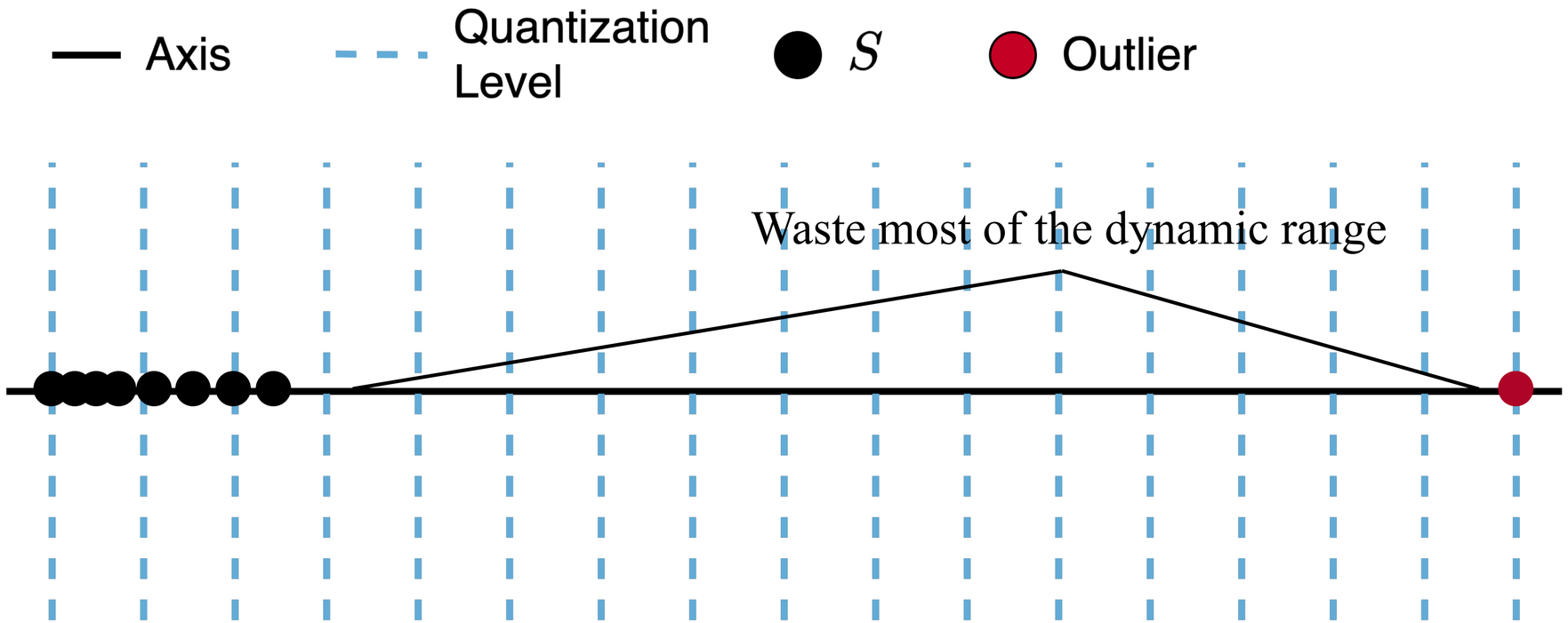
Assuming a set of data S sampled from uniform distribution, S can be quantized with INT4 perfectly.



Outliers: 1. It will squeeze the accuracy of other values.

What if we add an outlier which is much larger than value in S?

S loses most of its accuracy.



Outliers: 2. Outlier will amplify the quantization error of weight.

Let's use a simple model to estimate the factors of quantization error:

$$\Delta = A \cdot W - \underbrace{(A - \Delta A)}_{A_Q}(W - \underbrace{\Delta W}_{\text{Quant error of weight}}) = A \cdot \Delta W + W \cdot \Delta A - \Delta A \Delta W$$

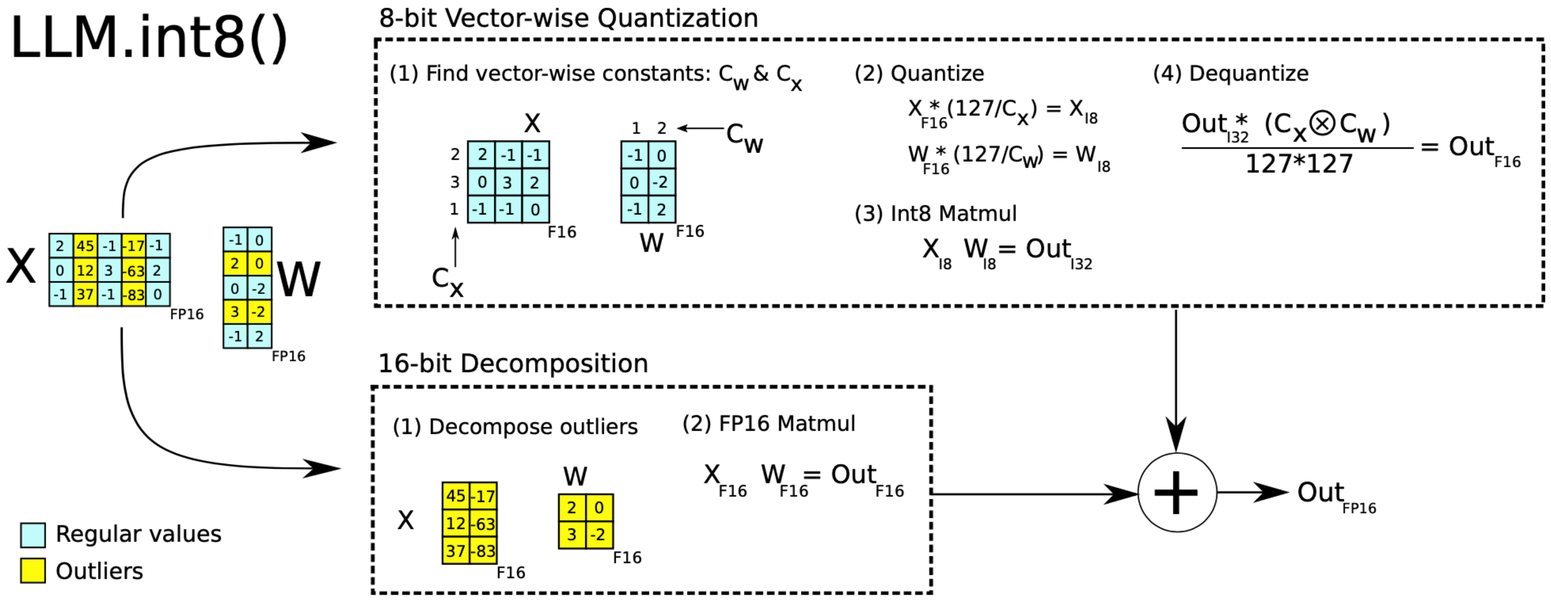
Outliers: 2. Outlier will amplify the quantization error of weight.

Let's use a simple model to estimate the factors of quantization error:

$$\Delta = A \cdot W - \underbrace{(A - \Delta A)}_{A_Q}(W - \underbrace{\Delta W}_{\text{Quant error of weight}}) = A \cdot \Delta W + W \cdot \Delta A - \Delta A \Delta W$$

Outliers contributes much larger error by amplifying the error of weight matrix, which becomes dominant in weight-only quantization setting.

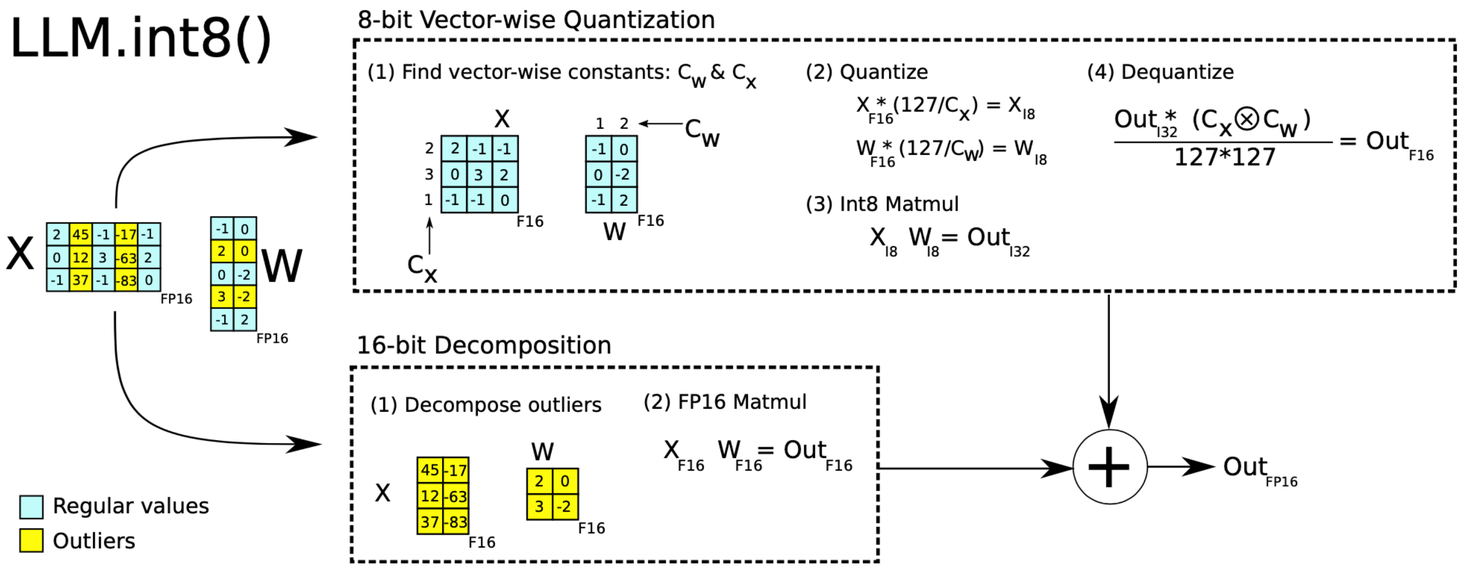
Mixed-precision: High precision for Outliers



- Treat outliers separately: Keep outliers in high precision while quantizing others to low-bit.
- Eliminate the influence of two aspects: Achieve better accuracy.

[1] Dettmers, T., Lewis, M., Belkada, Y., & Zettlemoyer, L. (2022). Llm.int8(): 8-bit Matrix Multiplication for Transformers at Scale. *arXiv [Cs.LG]*. Retrieved from http://arxiv.org/abs/2208.07339

Mixed-precision: High precision for Outliers



However, naive mixed-precision is inefficient for:

- Irregular memory access of decomposition
- Extra data movement&calculation of separate kernel

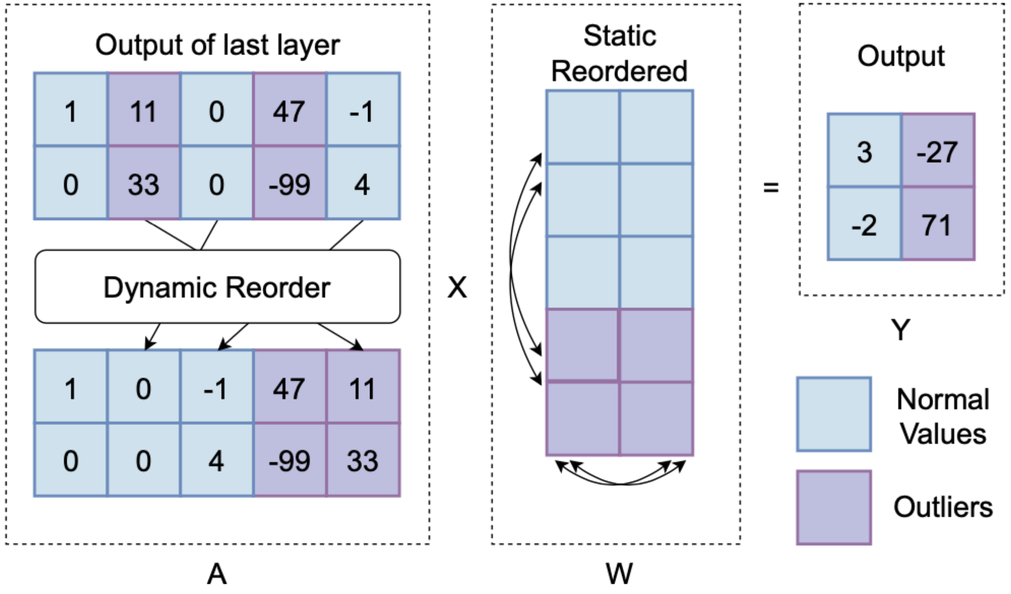
We need ordered input tensor like:

2	-1	-1	45	-17
0	3	2	12	-63
-1	-1	0	37	-83

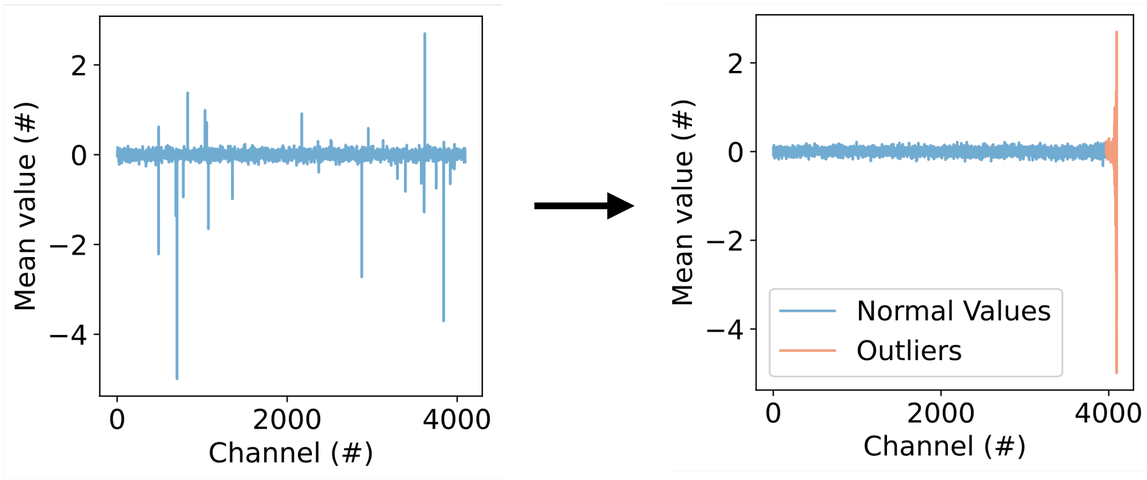
-1	0
0	-2
-1	2
2	0
3	-2

[1] Dettmers, T., Lewis, M., Belkada, Y., & Zettlemoyer, L. (2022). Llm.int8(): 8-bit Matrix Multiplication for Transformers at Scale. *arXiv [Cs.LG]*. Retrieved from http://arxiv.org/abs/2208.07339
[2] https://github.com/TimDettmers/bitsandbytes/blob/726f1470b3de8de024d30770a13959ebf1f9ad89/bitsandbytes/autograd/_functions.py#L399

Reorder Channels for Efficient Mixed-precision



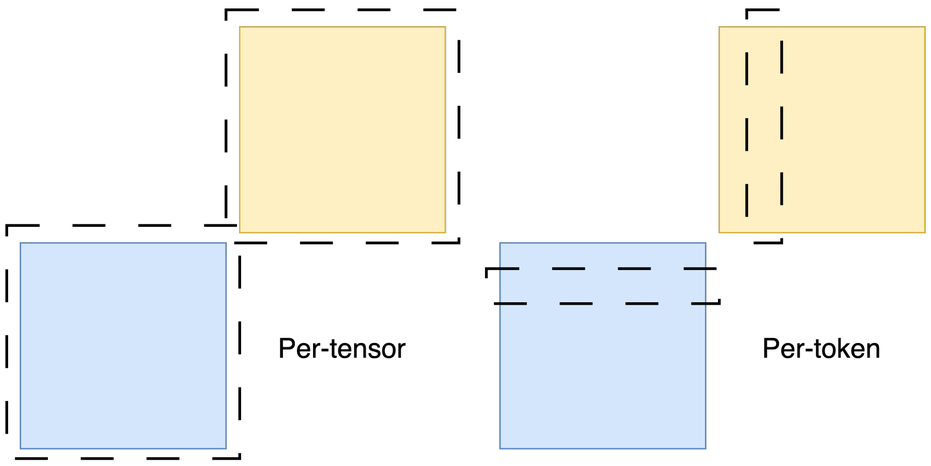
- Equivalent transformation of matrix multiplication.
- For correctness, reorder the columns of A and rows of W correspondingly.
- For the order of columns in Y, reorder the columns of W correspondingly.



[1] Yuan, Z., Niu, L., Liu, J., Liu, W., Wang, X., Shang, Y., … Wu, B. (2023). RPTQ: Reorder-based Post-training Quantization for Large Language Models. Retrieved from http://arxiv.org/abs/2304.01089

Fine-grained Quantization

- After picking out the outliers, accuracy loss is still large.
- Fundamental issue: limited expressivity of 4-bit, only 16 different values for large group number.

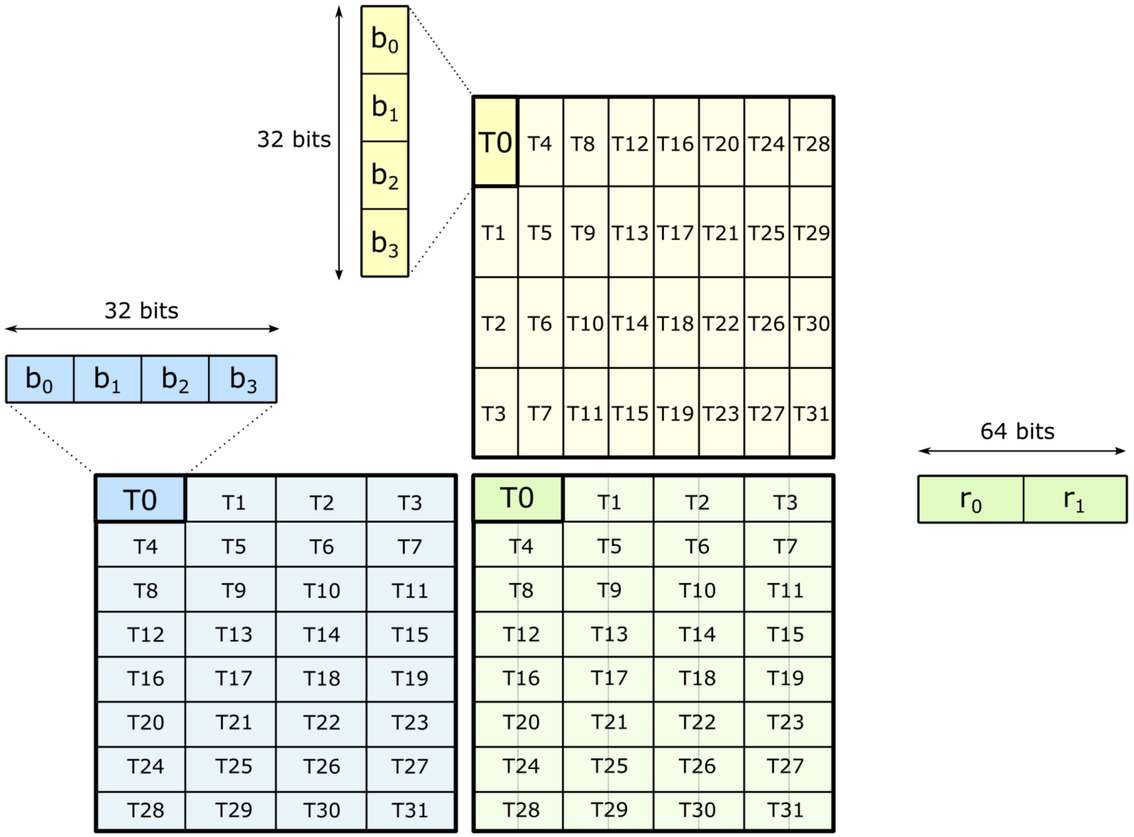


Typical granularity of quantization

- Assume b bits and g group size, average bits = $b + \frac{16}{g}$, number of values in each level: $\frac{g}{2^b}$
- For eg, from per-token $g = 4096$ to $g = 128$, resolution increases a lot while only adding small bits.

Fine-grained Quantization

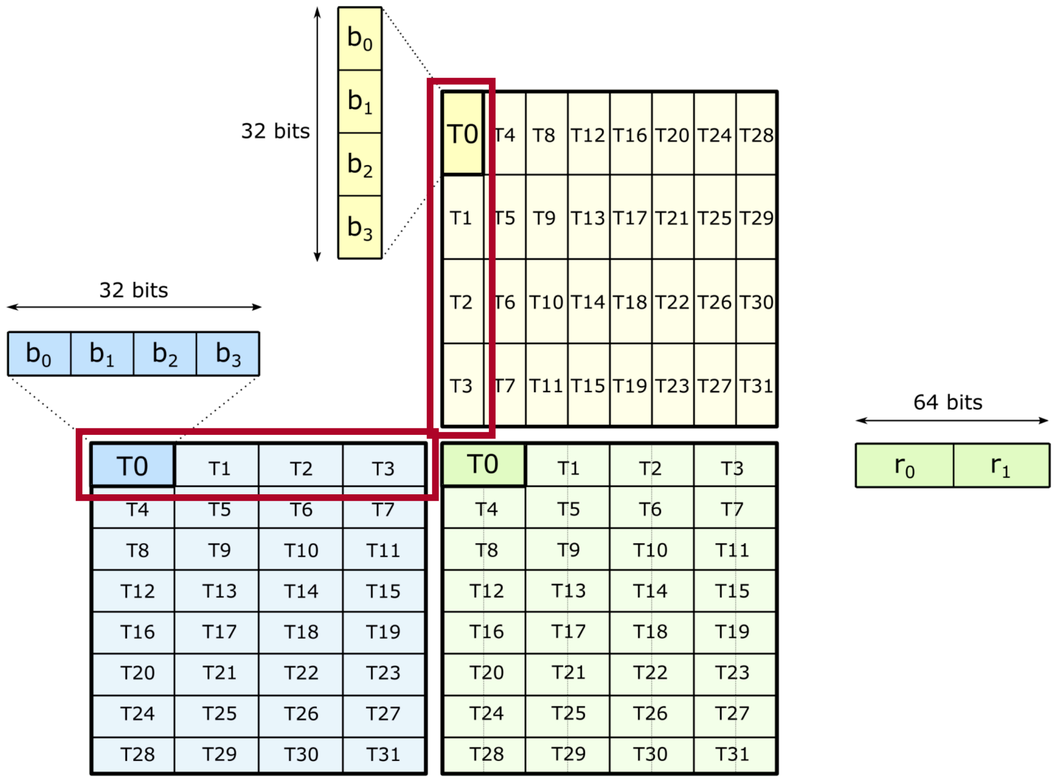
- Take a closer look at GPU: tensor core operation.
- 32 threads in a warp work collectively.



Detailed diagram of MMA with shape 8x8x128bits

[1] https://developer.download.nvidia.com/video/gputechconf/gtc/2020/presentations/s21745-developing-cuda-kernels-to-push-tensor-cores-to-the-absolute-limit-on-nvidia-a100.pdf

Fine-grained Quantization

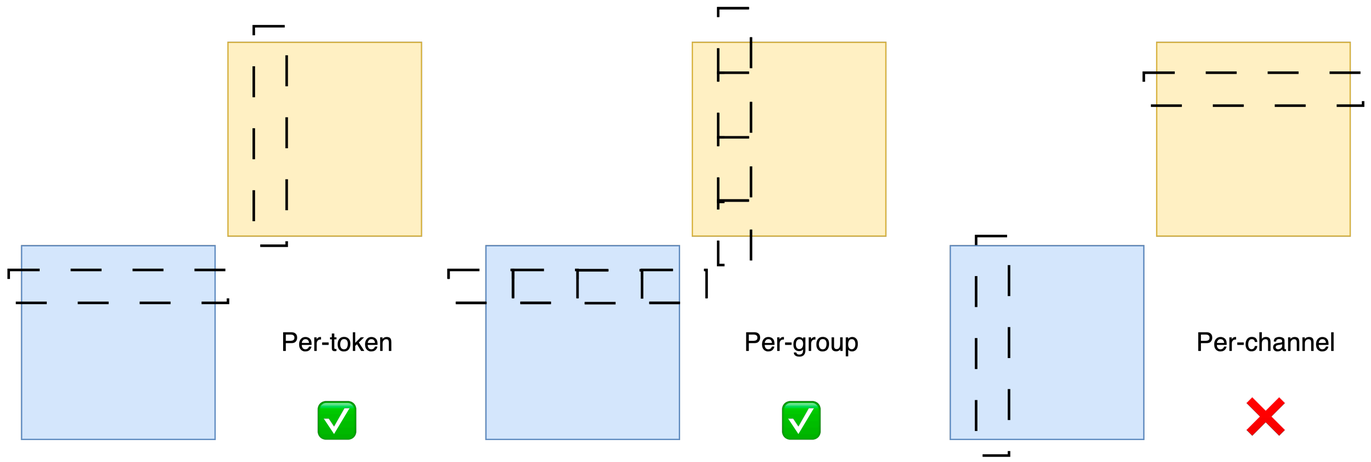


Scales can't be applied to the internal calculation, while accumulators can be manipulated.

↓

Group size should be multiple times of K shape of MMA

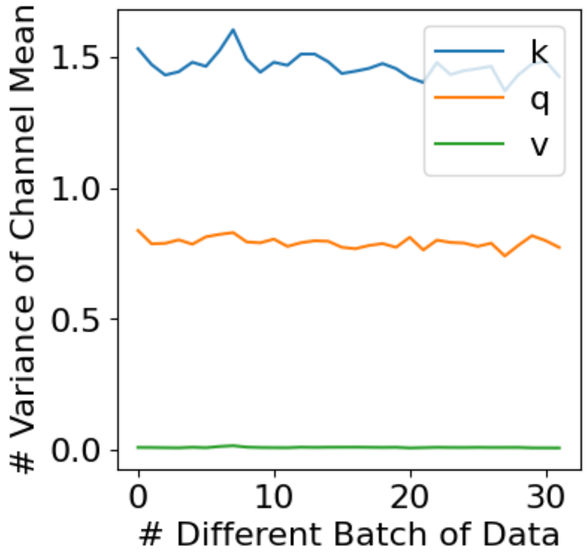
↓

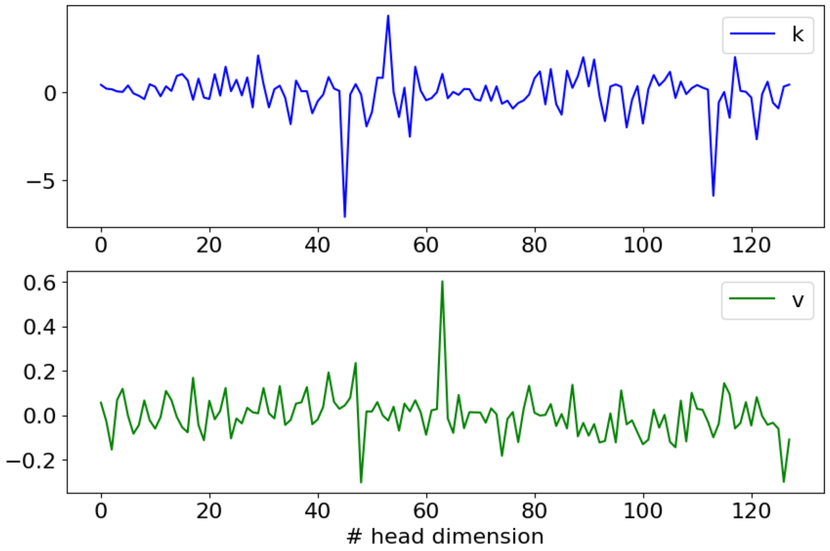


[1] https://developer.download.nvidia.com/video/gputechconf/gtc/2020/presentations/s21745-developing-cuda-kernels-to-push-tensor-cores-to-the-absolute-limit-on-nvidia-a100.pdf

KV-Quantization

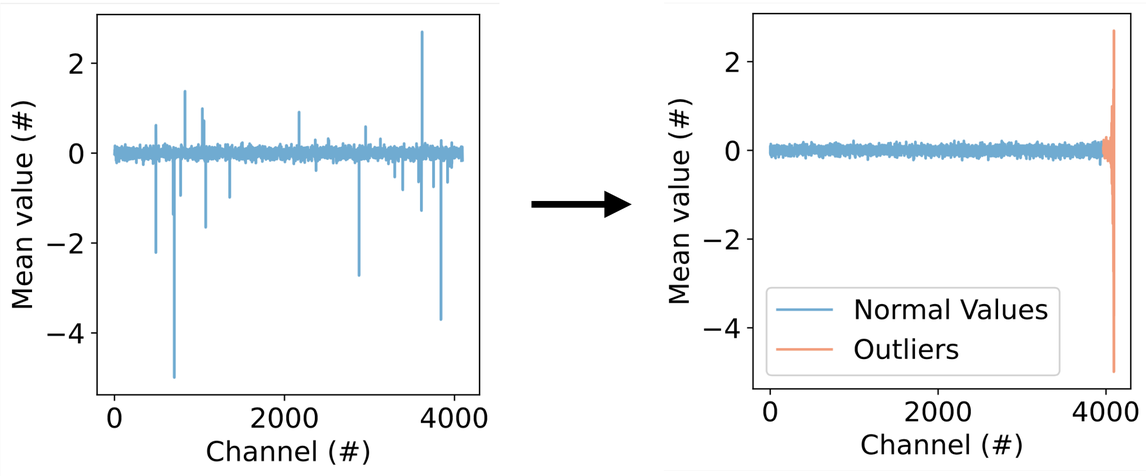
- Self-attention operator in decoding stage is highly memory-bound.
- Quantize KV optimizes for memory reduction and will boost the throughput.
- Different from weight-activation quantize setting, more similar to weight-only.
  We adopt asymmetric and group quantization (each head) for KV cache.



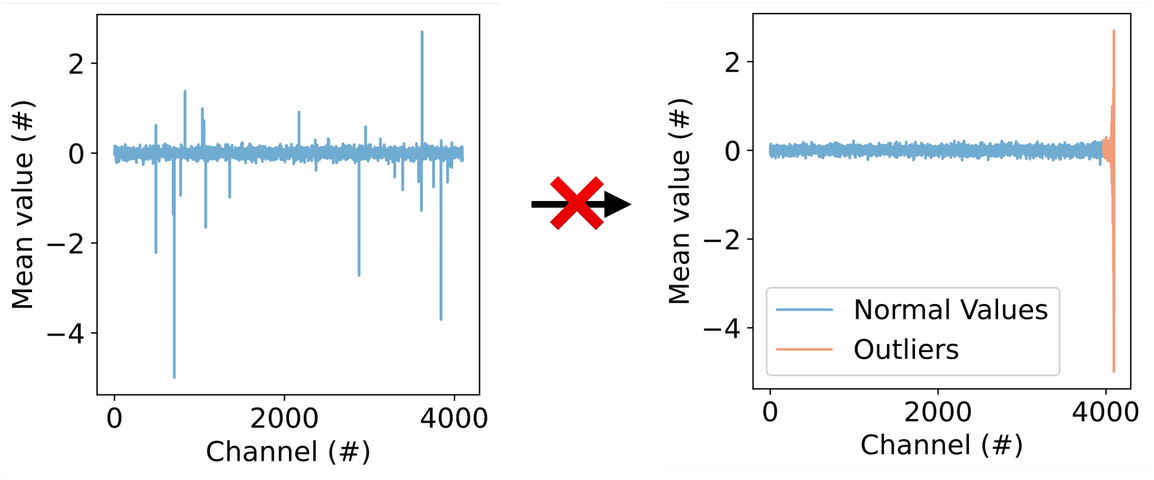


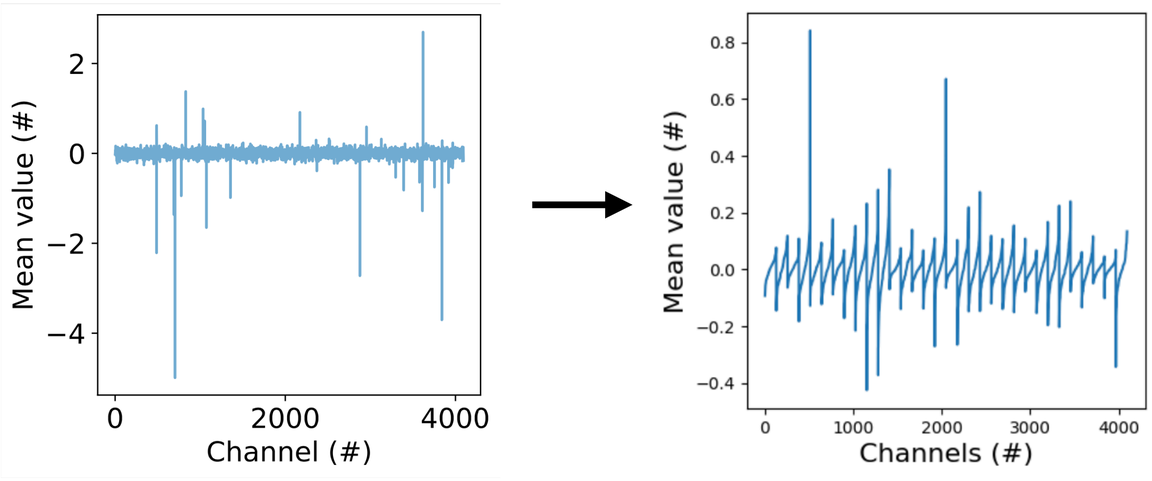
- V is more uniform and easy to quantize
- K will be smoothed by softmax, which makes it more robust.

KV-Quantization: Can we use reordering?



KV-Quantization: Can we use reordering?





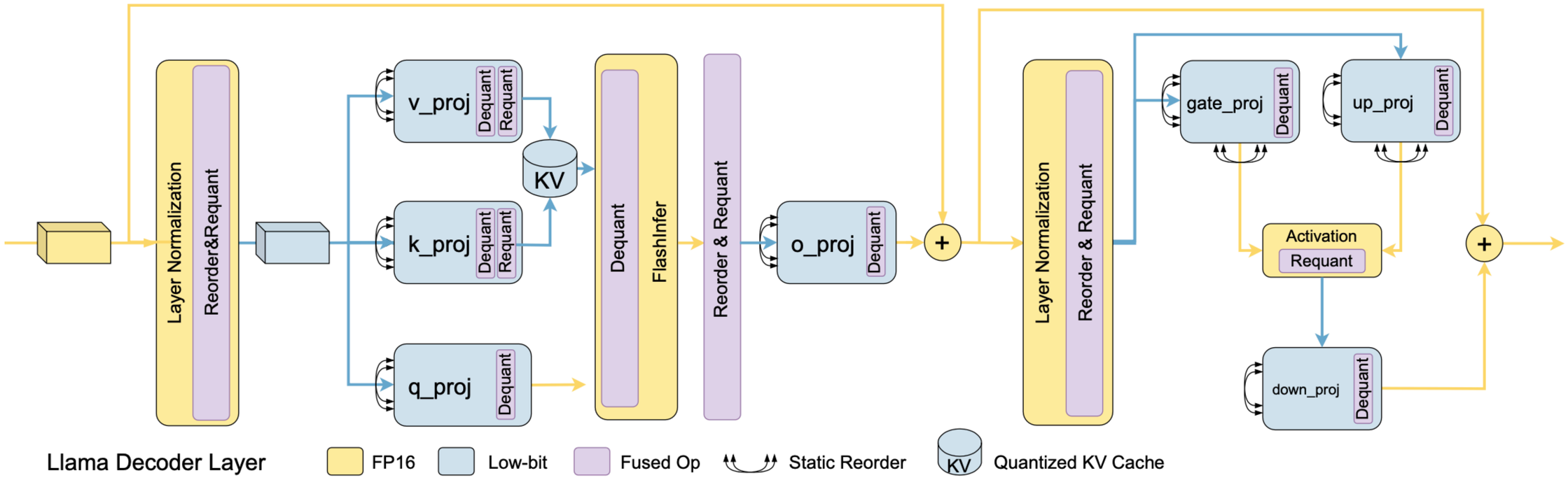
QK^T is batched matrix multiplication.

Reorder only works within the same head.

Specialized packed format of KV and corresponding dequantization operator.

Push to lower bits representation.

Implementation: Ideas about CUDA Kernel

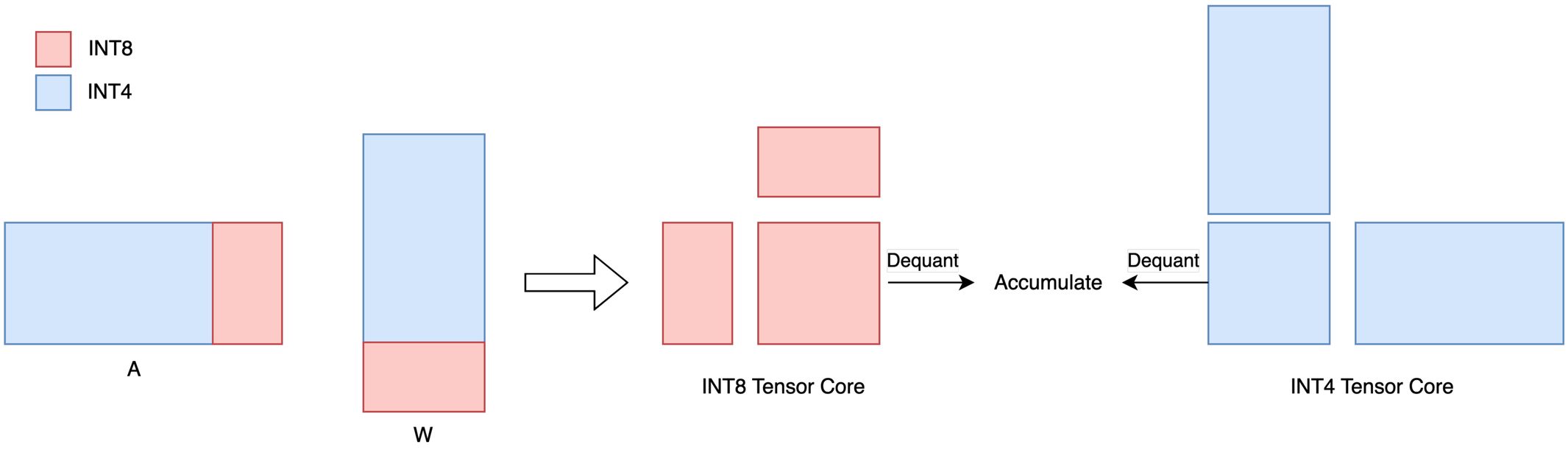


Workflow of Singe Decoder Block

Kernel Fusion is important for quantization operators.

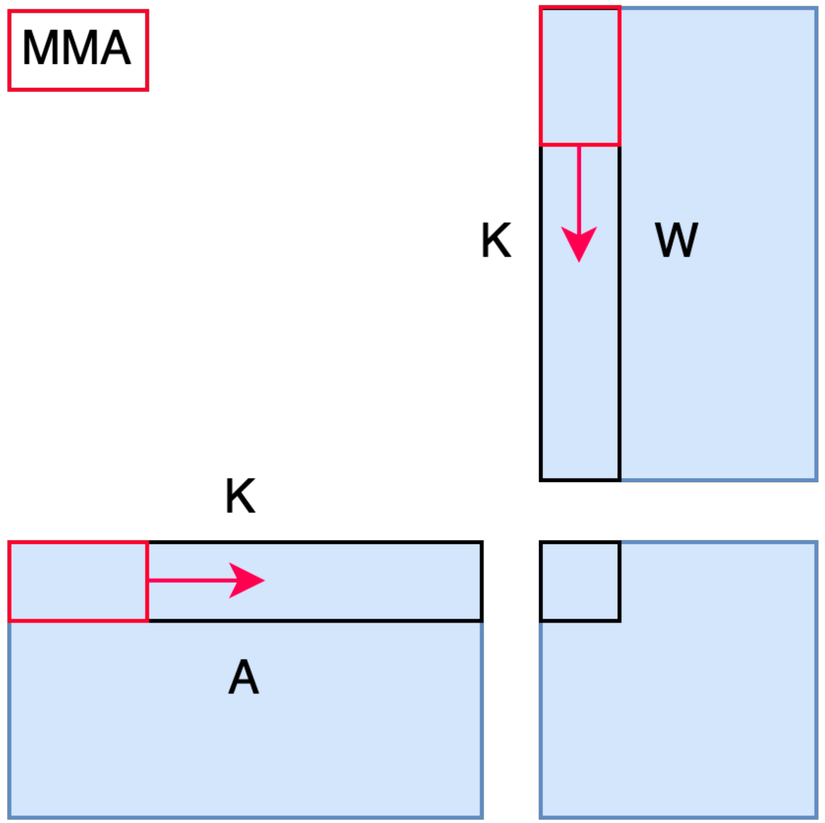
GEMM

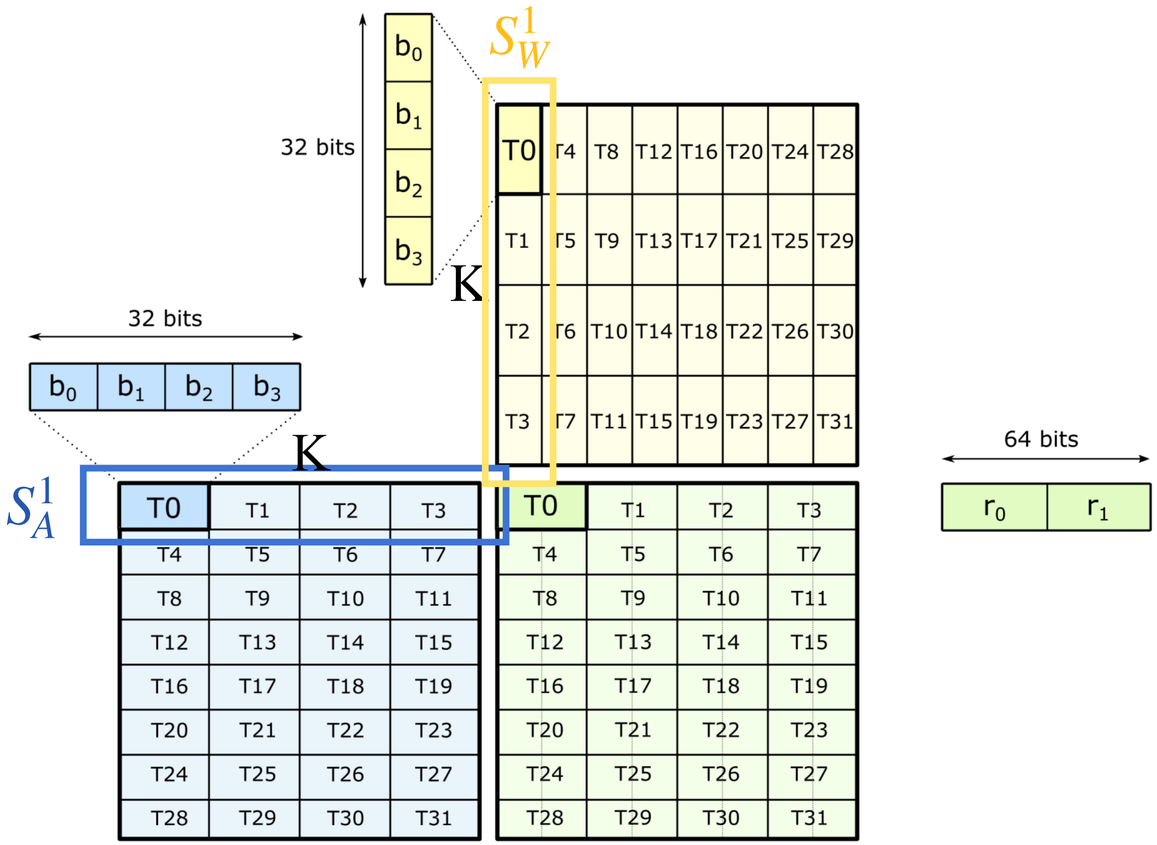
- Fused op: GEMM + group dequant + mixed-precision



GEMM

- Widely-used techniques for GEMM: Pipeline, double buffer, shmem swizzling -> 90% CUTLASS
- Dequant: take a close look

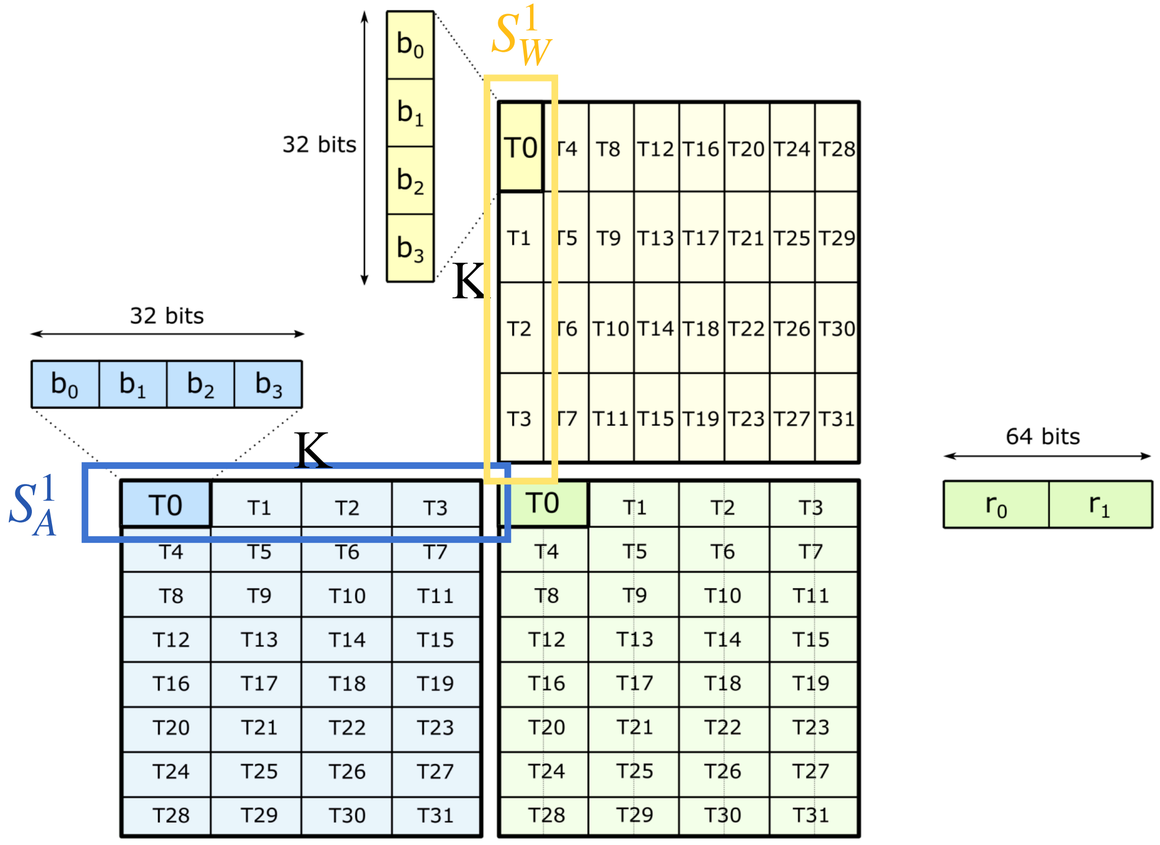




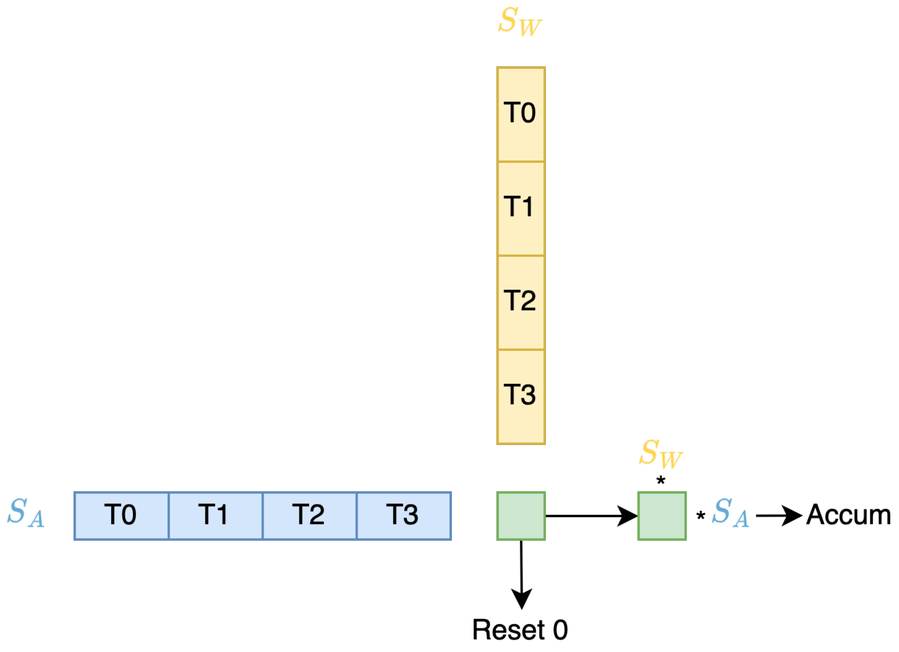
Detailed diagram of MMA with shape 8x8x128bits

GEMM

- Widely-used techniques for GEMM: Pipeline, double buffer, shmem swizzling -> 90% CUTLASS
- Dequant: take a close look

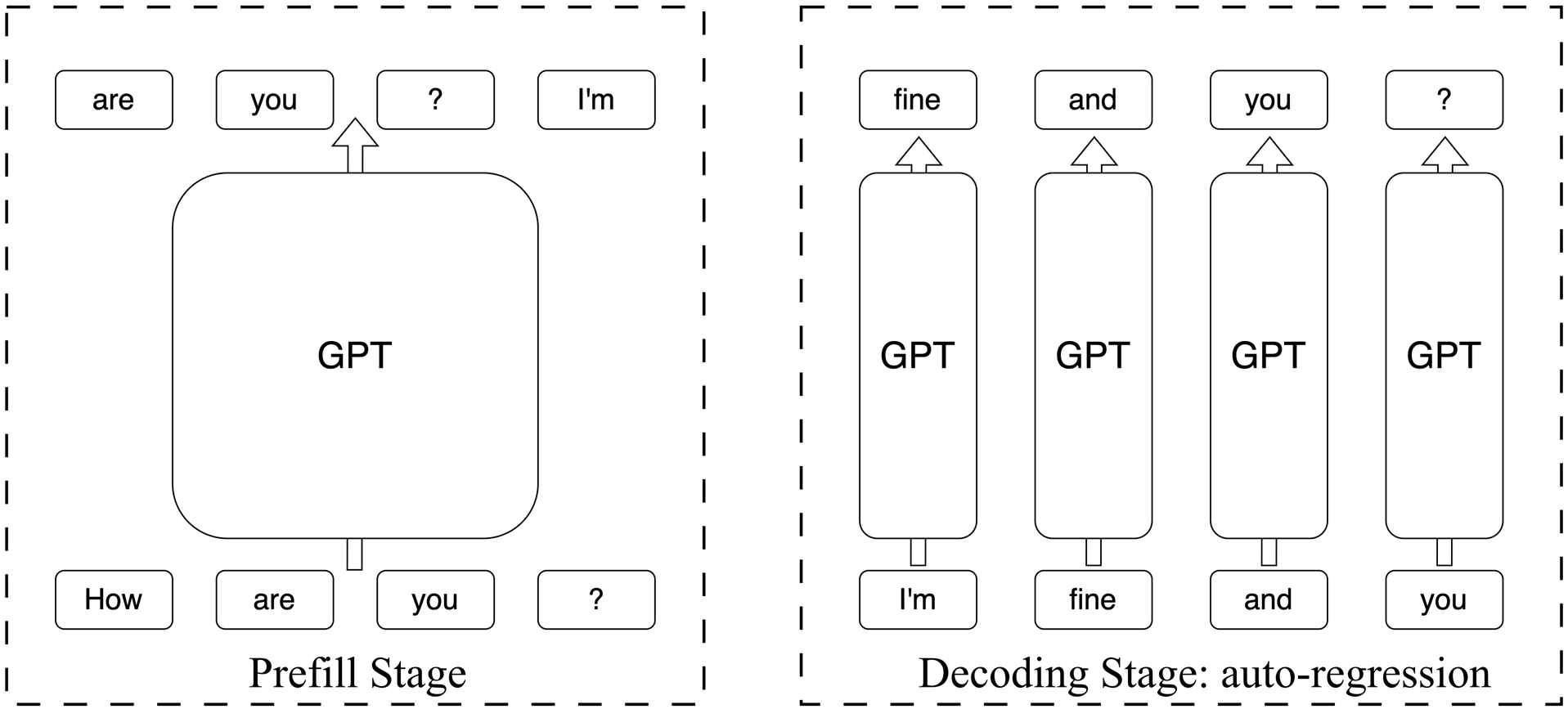


Detailed diagram of MMA with shape 8x8x128bits



Dequantization for each group's accumulation

Performance analysis: How can LLM get benefit from quantization?



Prefill stage: Operation characteristic

Initial Computation		Shape	FLOP	I/O	FLOP:I/O
Input					
X	Input for self attention	(b, s, h)			
Self-Attention					
$Q^{\ddagger}, K^{\ddagger}, V^{\ddagger}$	XW_Q, XW_K, XW_V	(b, s, h)	$O(bsh^2)$	$O(2bsh + h^2)$	$O(1/(1/h + 1/bs))$
$Q^{\dagger}, K^{\dagger}, V^{\dagger}$	Reshape $Q^{\ddagger}, K^{\ddagger}, V^{\ddagger}$	(b, s, n, d)			
Q, K, V	Transpose $Q^{\dagger}, K^{\dagger}, V^{\dagger}$	(b, n, s, d)			
K^T	Transpose K	(b, n, d, s)			
P	Softmax $\left(QK^T/\sqrt{d}\right)$	(b, n, s, s)	$O(bs^2nd)$	$O(2bsnd + bs^2n)$	$O(1/(1/d + 1/s))$
$A^{\ddagger}$	PV	(b, n, s, d)	$O(bs^2nd)$	$O(2bsnd + bs^2n)$	$O(1/(1/d + 1/s))$
$A^{\dagger}$	Transpose $A^{\ddagger}$	(b, s, n, d)			
A	Reshape $A^{\dagger}$	(b, s, h)			
Y	AW_O	(b, s, h)	$O(bsh^2)$	$O(2bsh + h^2)$	$O(1/(1/h + 1/bs))$
Feed-Forward Network					
Z	$\mathrm{ReLU}(YW_1)W_2$	(b, s, h)	$O(16bsh^2)$	$O(2bsh + 8h^2)$	$O(1/(1/h + 1/bs))$

- High arithmetic intensity
- Dense layer (QKV generation, O, FFN) can further benefit from batching

[1] https://le.qun.ch/en/blog/2023/05/13/transformer-batching/

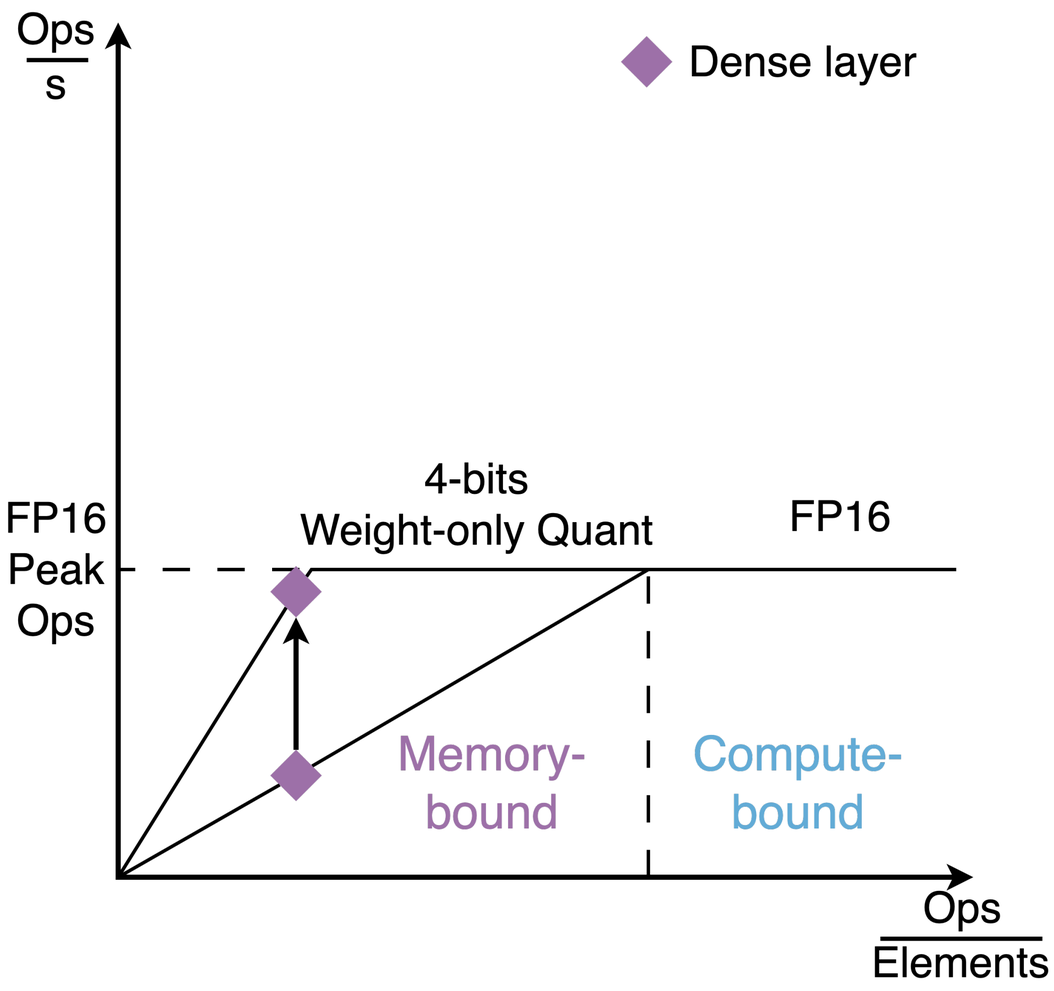
Decoding stage: Operation characteristic

Auto-Regression		Shape	FLOP	I/O	FLOP:I/O
Input					
x	Input for self attention	$(b, 1, h)$			
K_s, V_s	Key/Value cache from past	(b, n, s, d)			
Self-Attention					
$q^\ddagger, k^\ddagger, v^\ddagger$	xW_Q, xW_K, xW_V	$(b, 1, h)$	$O(bh^2)$	$O(2bh + h^2)$	$O(1/(1/h + 1/b))$
$q^\dagger, k^\dagger, v^\dagger$	Reshape $q^\ddagger, k^\ddagger, v^\ddagger$	$(b, 1, n, d)$			
q, k, v	Transpose $q^\dagger, k^\dagger, v^\dagger$	$(b, n, 1, d)$			
K, V	$\text{concat}(K_s, k), \text{concat}(V_s, v)$	$(b, n, s+1, d)$			
K^T	Transpose K	$(b, n, d, s+1)$			
p	Softmax $\left(qK^T/\sqrt{d}\right)$	$(b, n, 1, s+1)$	$O(bsnd)$	$O(bsn + bsnd + bnd)$	$O(1/(1 + 1/d + 1/s))$
$a^\ddagger$	pV	$(b, n, 1, d)$	$O(bsnd)$	$O(bsn + bsnd + bnd)$	$O(1/(1 + 1/d + 1/s))$
$a^\dagger$	Transpose $a^\ddagger$	$(b, 1, n, d)$			
a	Reshape $a^\dagger$	$(b, 1, h)$			
y	aW_O	$(b, 1, h)$	$O(bh^2)$	$O(2bh + h^2)$	$O(1/(1/h + 1/b))$
Feed-Forward Network					
z	$\text{ReLU}(yW_1)W_2$	$(b, 1, h)$	$O(16bh^2)$	$O(2bh + 8h^2)$	$O(1/(1/h + 1/b))$

- Low arithmetic intensity
- Dense layer's arithmetic intensity is dominant by batch size
- Attention op has extremely low intensity

[1] https://le.qun.ch/en/blog/2023/05/13/transformer-batching/

Weight-only quantization

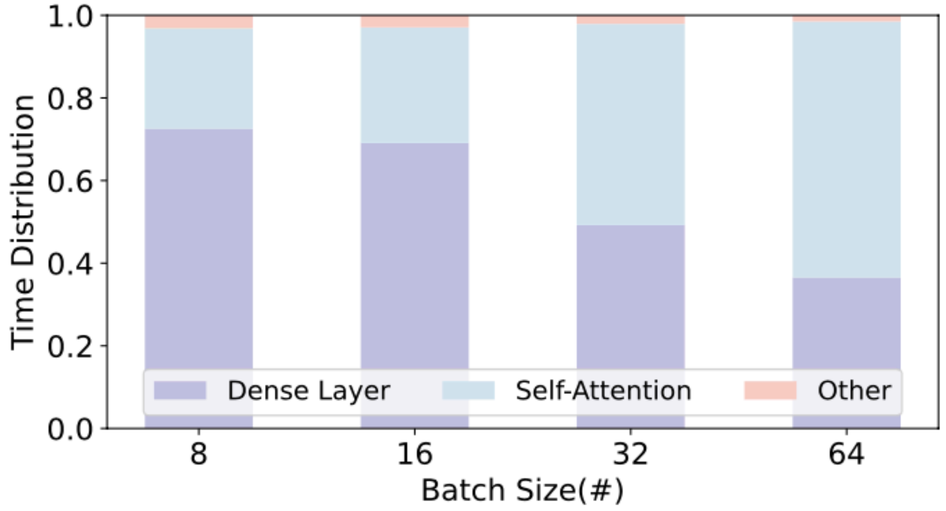


- Use roofline model to estimate.
- Setting: single user inference
- No batching: memory-bound

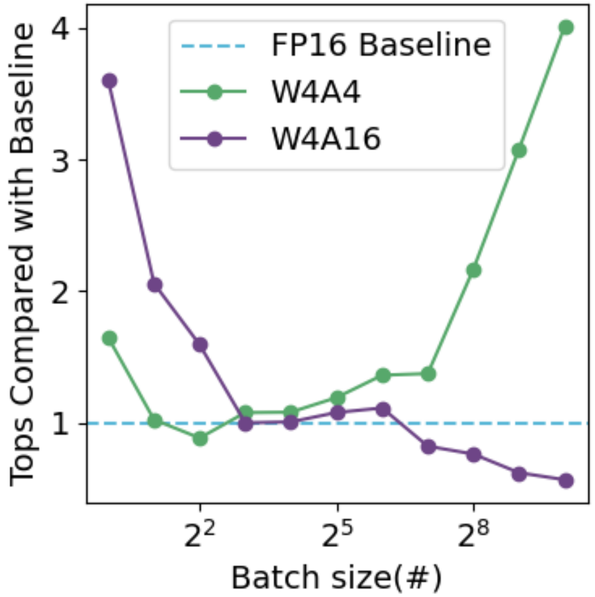
Weight-only quantization works good!

Weight-only quantization fails

- Setting: multi-user serving
- Batching is widely used for increasing throughput.



Shape: Bsz x 4096 x 4096

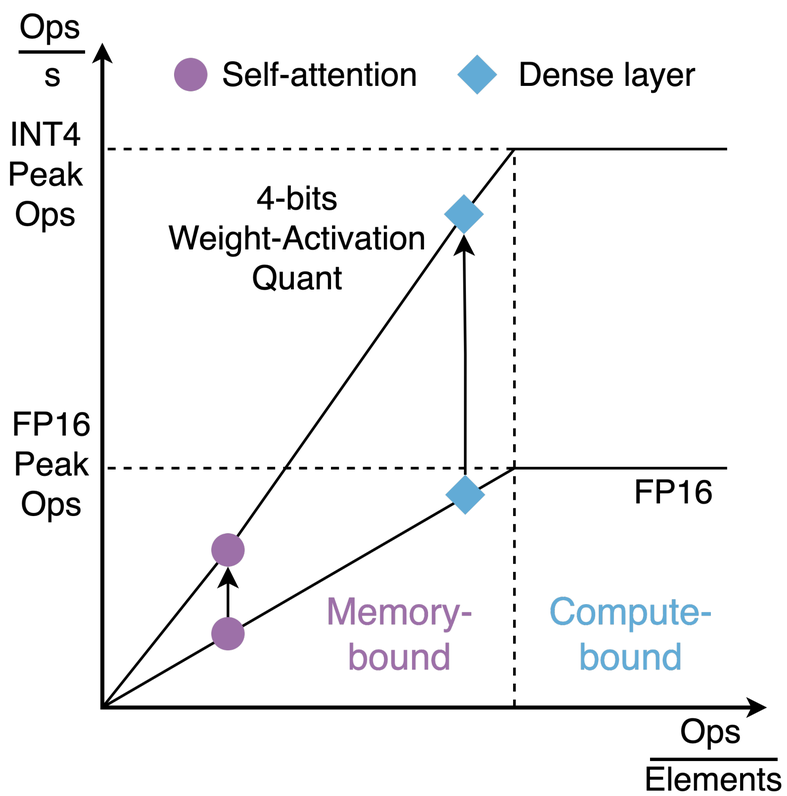


Self-attention $(softmax(\frac{QK^T}{\sqrt{d}})V)$ becomes bottleneck

Speedup drops sharply as arithmetic increases

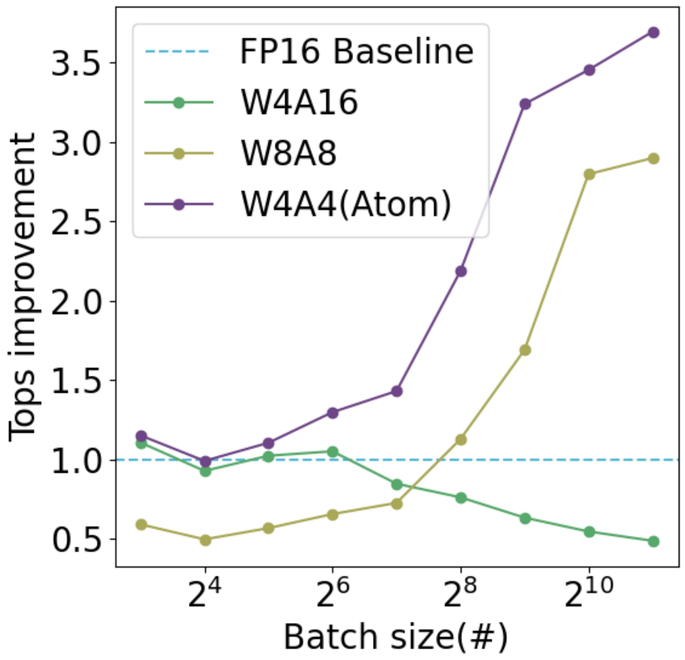
[1] Time profiled by nsys profiler. [2] W4A4: Max throughput result of AWQ GEMM and GEMV kernel.

Weight-activation quantization works

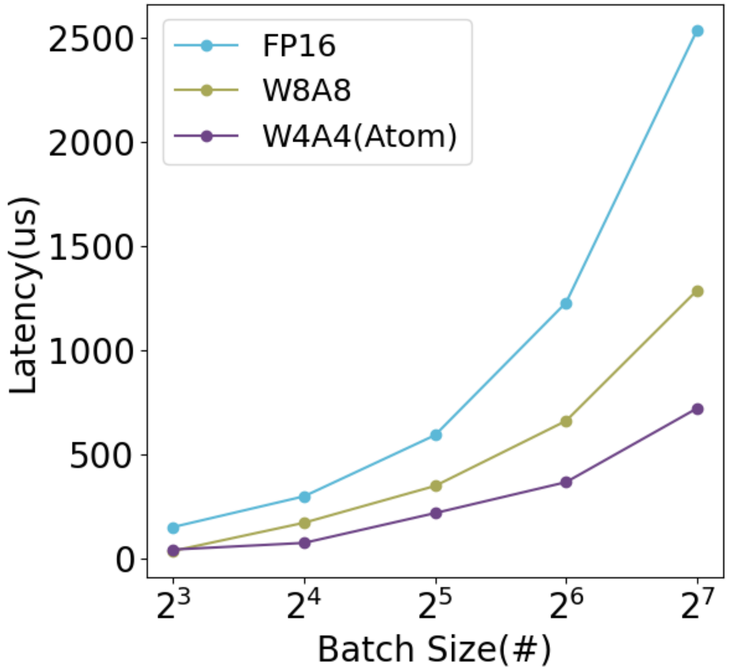


Theoretical analyze by roofline

GEMM: bsz x 4096 x 4096



Self-attention: seqlen=1024



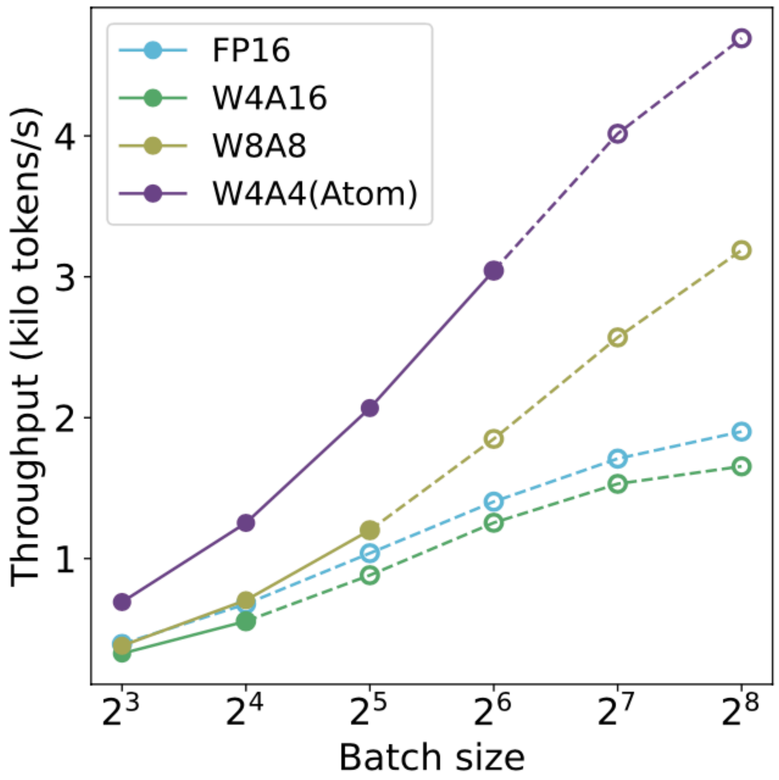
Practical evaluation of kernels
RTX4090 CUDA 12.1

[1] Time profiled by nsys profiler.

End-to-end evaluation

- Integrated into a serving framework, Punica
- Compared with SmoothQuant, AWQ and FP16. All shares the same backbone kernels.
- Sample requests length from ShareGPT and do continuous batching from Orca.
- Test on a single 24GB RTX4090.

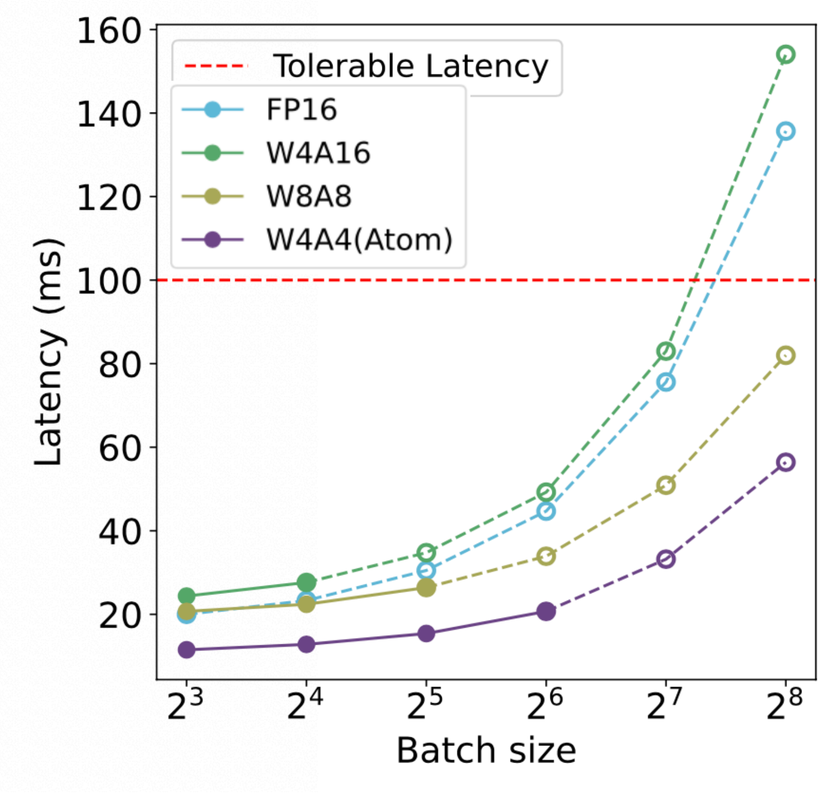
- Atom achieves better throughput at same batch size due to the efficient quantization mechanism
- Atom achieves 2.53x throughput compared to SmoothQuant, by having larger batch size



[1] Chen, L., Ye, Z., Wu, Y., Zhuo, D., Ceze, L., & Krishnamurthy, A. (2023). Punica: Multi-Tenant LoRA Serving. arXiv [Cs.DC]. Retrieved from http://arxiv.org/abs/2310.18547
[2] Yu, G.-I., Jeong, J. S., Kim, G.-W., Kim, S., & Chun, B.-G. (2022, July). Orca: A Distributed Serving System for Transformer-Based Generative Models.

End-to-end evaluation

- Integrated into a serving framework, Punica
- Compared with SmoothQuant, AWQ and FP16. All shares the same backbone kernels.
- Sample requests length from ShareGPT and do continuous batching from Orca.
- Test on a single 24GB RTX4090.



- Atom maintains lower latency at same batch size due to efficient quantization mechanism.
- Even at 256 batch size, Atom still maintains decent latency.

[1] Chen, L., Ye, Z., Wu, Y., Zhuo, D., Ceze, L., & Krishnamurthy, A. (2023). Punica: Multi-Tenant LoRA Serving. arXiv [Cs.DC]. Retrieved from http://arxiv.org/abs/2310.18547
[2] Yu, G.-I., Jeong, J. S., Kim, G.-W., Kim, S., & Chun, B.-G. (2022, July). Orca: A Distributed Serving System for Transformer-Based Generative Models.

Discussion & Future works

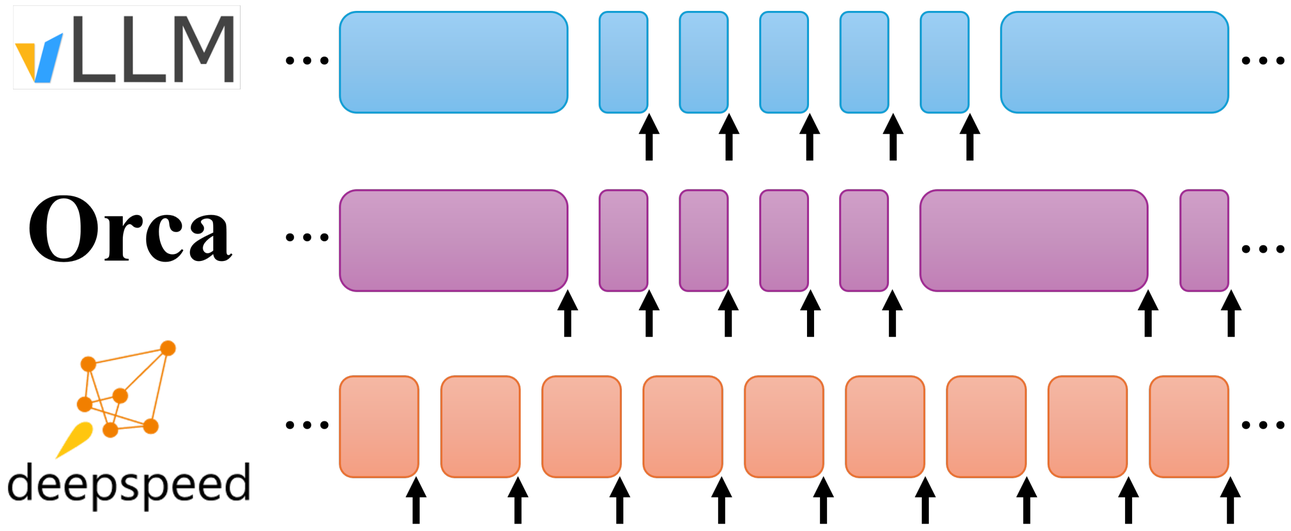
Discussion & Future works

1. Atom can do better at larger batch size. But how to design a serving framework with large batch size?

   Memory capacity is the challenge: one request's KV in Llama-7b consumes 0.5GB (32*1024*4096*2*2/1024/1024)

   Interesting observation: prefill request is more "cost-efficient" than decoding request.

   DeepSpeed-FastGen: Decompose prefill requests and batch them with decoding requests.



[1] https://github.com/microsoft/DeepSpeed/tree/master/blogs/deepspeed-fastgen#3-dynamic-splitfuse-a-novel-prompt-and-generation-composition-strategy

Discussion & Future works

2. Self-attention becomes new bottleneck. How to boost the speed of it?

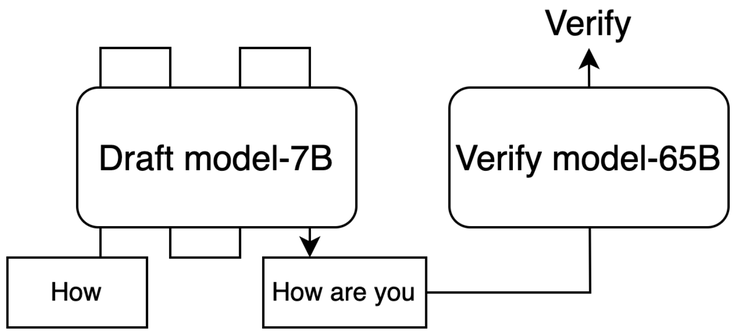
   Straightforward: quantize, prune, …

   But can we do batching on self-attention?

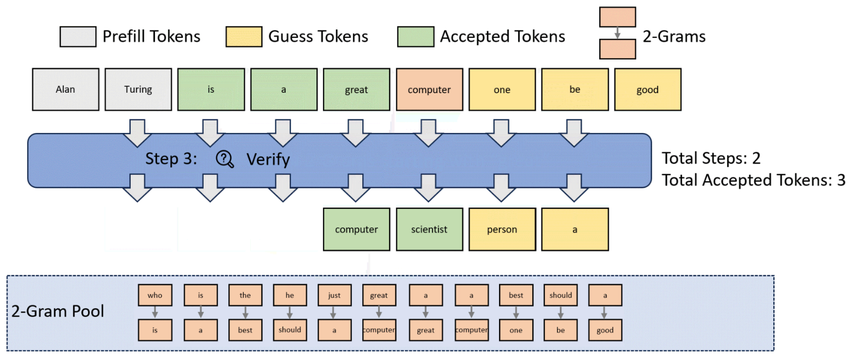
Discussion & Future works

2. Self-attention becomes new bottleneck. How to boost the speed of it?

Temporal batching: batch several time steps of single request

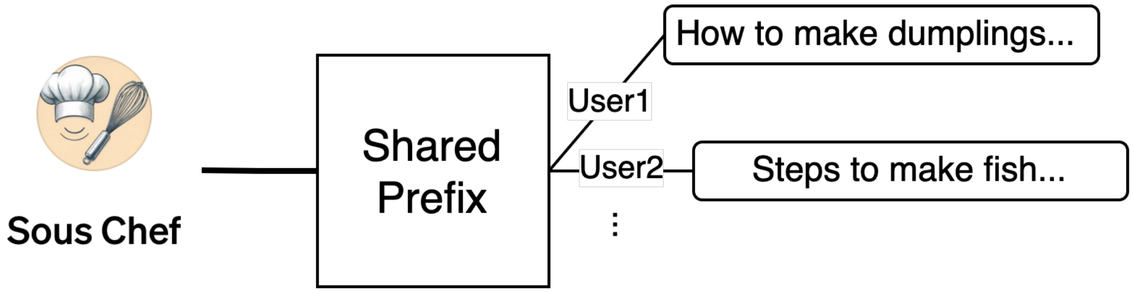


Speculative Decoding



Lookahead Decoding

Spacial batching: batch different requests with shared-prefix (like GPTs)



[1] Fu, Y., Bailis, P., Stoica, I., & Zhang, H. (2023, November). Breaking the Sequential Dependency of LLM Inference Using Lookahead Decoding.

Discussion & Future works

3.New low-bit data type support?

FP8

FP4: LLM.FP4

NVIDIA log representation

$$v = -1^{s} 2^{ei.ef}$$

[1] Liu, S.-Y., Liu, Z., Huang, X., Dong, P., & Cheng, K.-T. (2023). LLM-FP4: 4-Bit Floating-Point Quantized Transformers. *arXiv [Cs.CL]*. Retrieved from http://arxiv.org/abs/2310.16836
[2] Bill Dally, NVIDIA, HC2023-K2: Hardware for Deep Learning

Thanks for your attention! Q&A

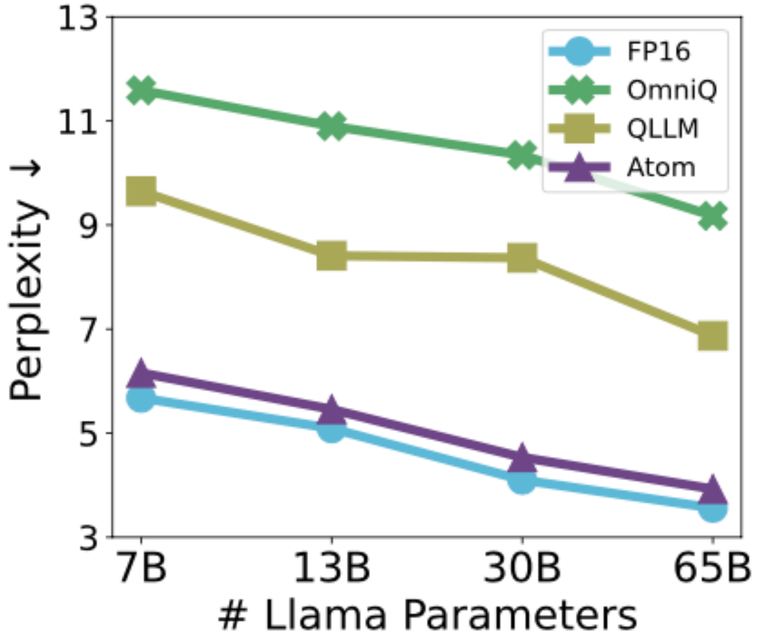


Table 2. Perplexity of quantized Llama models on WikiText2, PTB and C4 dataset.

Size	Bits	Method	Perplexity ↓ WikiText2	PTB	C4
7B	FP16	-	5.68	8.80	7.08
	W4A4	SmoothQuant	22.62	40.69	31.21
		OmniQuant	11.59	20.65	14.96
		QLLM	9.65	-	12.29
		Atom	**6.16**	**9.62**	**7.69**
	W3A3	SmoothQuant	2.7e4	3.5e4	2.6e4
		OmniQuant	3.4e3	7.5e3	6.3e3
		Atom	**10.67**	**18.10**	**13.65**
30B	FP16	-	4.10	7.30	5.98
	W4A4	SmoothQuant	109.85	142.34	87.06
		OmniQuant	10.34	14.91	12.49
		QLLM	8.37	-	11.51
		Atom	**4.54**	**7.67**	**6.35**
	W3A3	SmoothQuant	1.5e4	1.6e4	1.5e4
		Atom	**6.99**	**11.07**	**9.14**

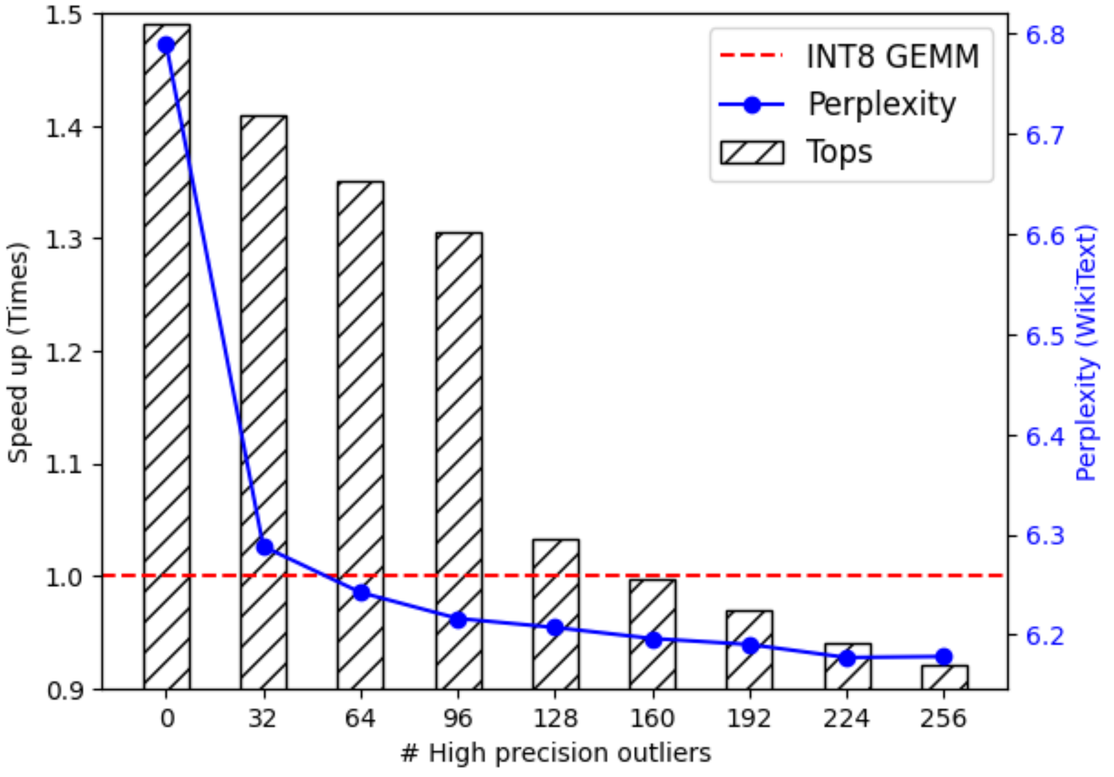
Size	Bits	Method	Perplexity ↓ WikiText2	PTB	C4
13B	FP16	-	5.09	8.07	6.61
	W4A4	SmoothQuant	33.98	73.83	41.53
		OmniQuant	10.90	18.03	13.78
		QLLM	8.41	-	10.58
		Atom	**5.46**	**8.60**	**7.03**
	W3A3	SmoothQuant	1.3e4	1.6e4	1.5e4
		OmniQuant	7.2e3	1.6e4	1.3e4
		Atom	**8.40**	**14.31**	**10.76**
65B	FP16	-	3.56	6.91	5.62
	W4A4	SmoothQuant	88.89	278.76	283.80
		OmniQuant	9.18	16.18	11.31
		QLLM	6.87	-	8.98
		Atom	**3.93**	**7.54**	**5.92**
	W3A3	SmoothQuant	6.6e8	3.7e8	4.4e8
		Atom	**5.79**	**9.30**	**7.84**

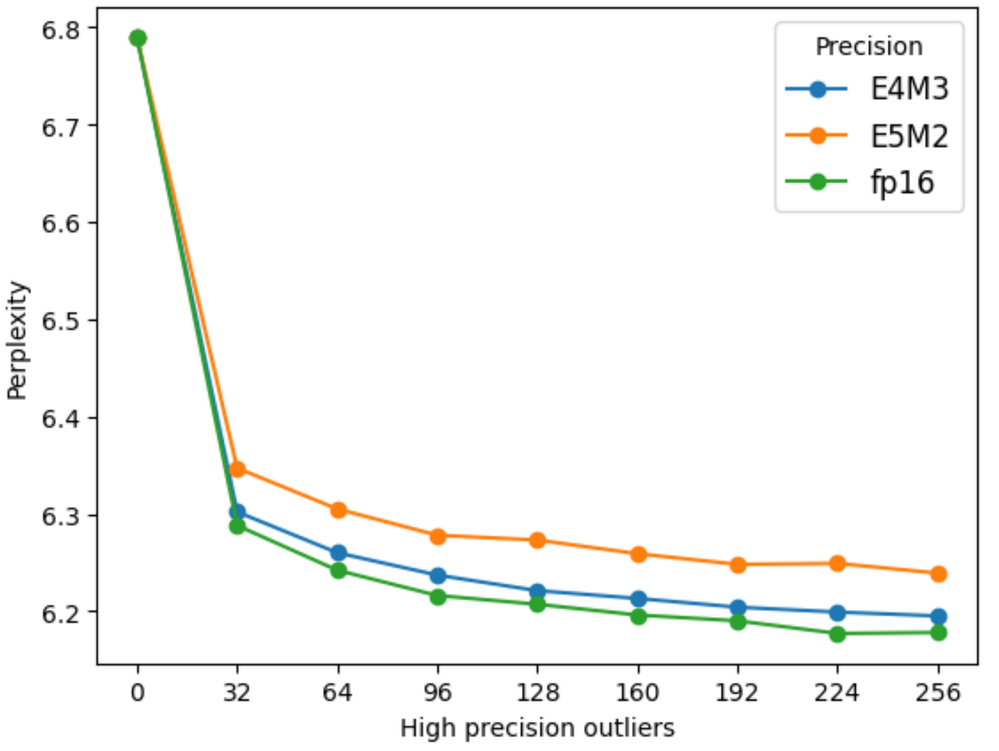
Table 4. Ablation study on different quantization techniques used in Atom. The model used in this table is Llama-7B.

Quantization method	**WikiText2 PPL↓**
FP16 baseline	5.68
W4A4 RTN	2315.52
+ Keeping 128 outliers in FP16	11.34 (2304.2↓)
+ Quantizing outliers to INT8	11.39 (0.05↑)
+ Group size 128	6.22 (5.17↓)
+ Clipping	6.13 (0.09↓)
+ GPTQ	6.04 (0.09↓)
+ Quantizing KV-cache to INT4	6.16 (0.12↑)

Table 1. **Zero-shot accuracy** of quantized Llama models on six common sense tasks.

Llama	#Bits	Method	Zero-shot Accuracy ↑						
			PIQA	ARC-e	ARC-c	BoolQ	HellaSwag	Winogrande	Avg.
7B	FP16	-	77.42	52.61	41.47	73.12	72.98	67.01	64.10
	W4A4	SmoothQuant	63.11	40.03	31.57	58.47	43.38	52.80	48.23
		OmniQuant	66.15	45.20	31.14	63.51	56.44	53.43	52.65
		QLLM	68.77	45.20	31.14	-	57.43	56.67	51.84
		Atom	**75.57**	**51.30**	**37.88**	**72.05**	**69.84**	**62.35**	**61.50**
	W3A3	SmoothQuant	48.69	25.97	28.16	45.26	26.02	49.57	37.28
		OmniQuant	49.78	27.19	27.22	37.86	25.64	49.96	36.28
		Atom	66.10	42.21	31.40	62.48	52.90	52.25	51.22
13B	FP16	-	79.11	59.85	44.54	68.47	76.22	70.09	66.38
	W4A4	SmoothQuant	64.47	41.75	30.89	62.29	46.68	51.70	49.63
		OmniQuant	69.69	47.39	33.10	62.84	58.96	55.80	54.63
		QLLM	71.38	47.60	34.30	-	63.70	59.43	55.28
		Atom	**77.37**	**57.66**	**43.26**	**66.82**	**73.71**	**68.59**	**64.57**
	W3A3	SmoothQuant	47.99	26.30	27.65	46.91	25.65	49.64	37.36
		OmniQuant	50.22	26.77	27.82	37.83	25.77	51.07	36.58
		Atom	70.73	48.06	33.96	63.70	62.64	57.85	56.16
30B	FP16	-	80.20	58.92	45.31	68.38	79.23	72.69	67.46
	W4A4	SmoothQuant	59.30	36.74	28.58	59.97	34.84	49.96	44.90
		OmniQuant	71.21	49.45	34.47	65.33	64.65	59.19	57.38
		QLLM	73.83	50.67	38.40	-	67.91	58.56	57.87
		Atom	**78.73**	**58.92**	**45.82**	**68.47**	**77.40**	**73.09**	**67.07**
	W3A3	SmoothQuant	49.46	27.53	28.16	39.42	26.05	51.38	37.00
		Atom	72.47	49.54	37.80	65.75	66.99	60.14	58.78
65B	FP16	-	80.79	58.71	46.24	82.29	80.72	77.50	71.04
	W4A4	SmoothQuant	60.72	38.80	30.29	57.61	36.81	53.43	46.28
		OmniQuant	71.81	48.02	35.92	73.27	66.81	59.51	59.22
		QLLM	73.56	52.06	39.68	-	70.94	62.90	59.83
		Atom	**80.41**	**58.12**	**45.22**	**82.02**	**79.10**	**72.53**	**69.57**
	W3A3	SmoothQuant	49.56	26.64	29.10	42.97	26.05	51.14	37.58
		Atom	75.84	51.43	41.30	74.07	72.22	64.33	63.20


INT8 GEMM
Perplexity
Tops
Speed up (Times)
Perplexity (WikiText)
High precision outliers
1.5
1.4
1.3
1.2
1.1
1.0
0.9
6.8
6.7
6.6
6.5
6.4
6.3
6.2
0
32
64
96
128
160
192
224
256


Precision
E4M3
E5M2
fp16
Perplexity
High precision outliers
6.8
6.7
6.6
6.5
6.4
6.3
6.2
0
32
64
96
128
160
192
224
256