Speeding up LLM and GEMM

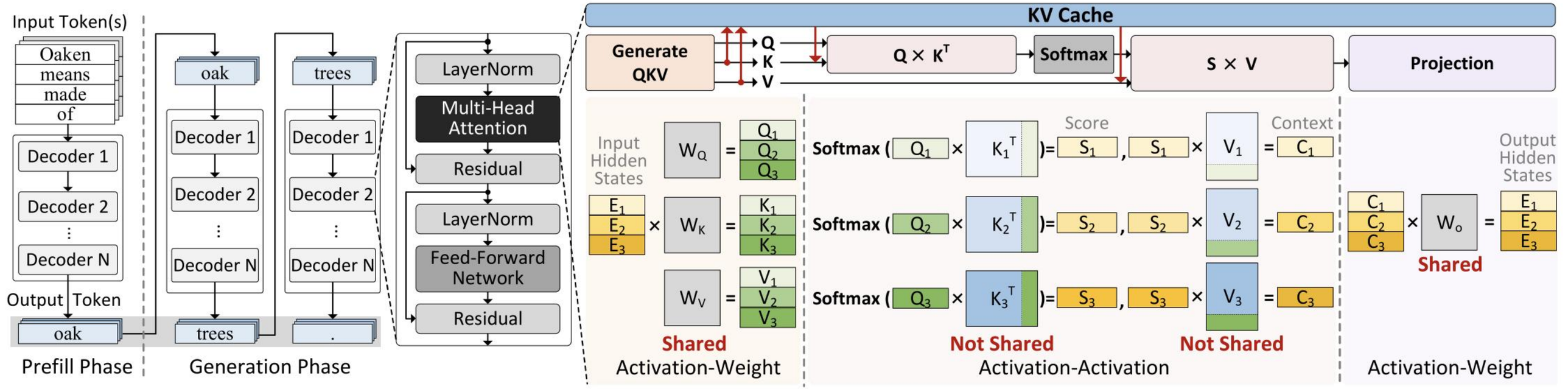
Two papers from ISCA 2025

Wenxuan Miao 6.27

Oaken

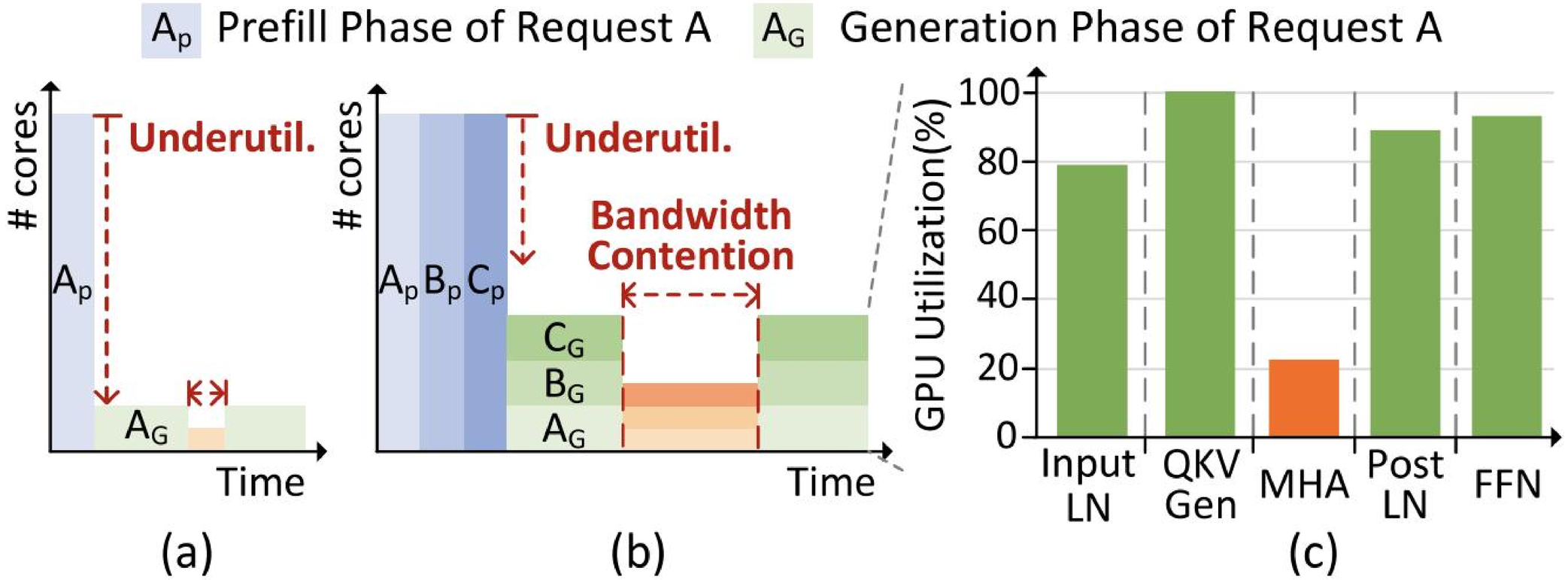
Fast and Efficient LLM Serving with Online-Offline Hybrid KV Cache Quantization

Overview of KV cache



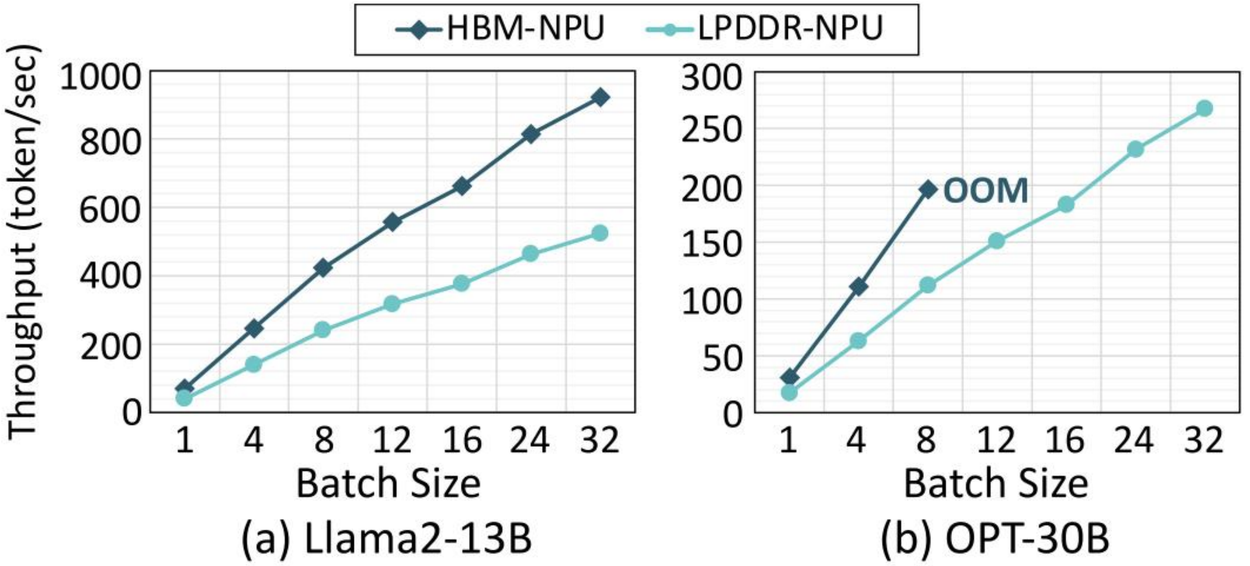
KV cache is widely used in LLM inference. However, in multi-request scenarios, the inability to share KV cache across different requests results in significant memory overhead.

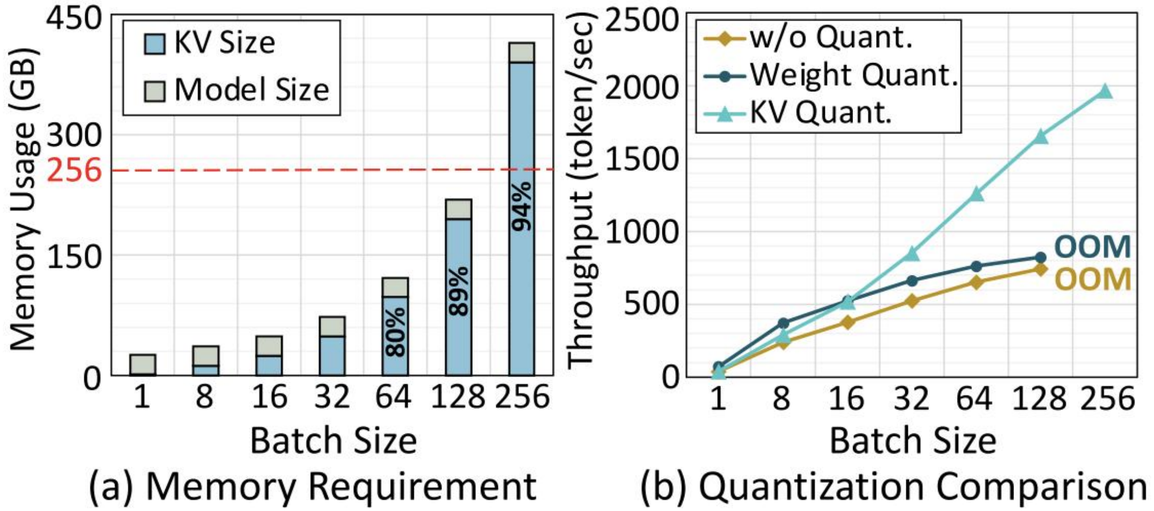
Characteristic analysis of LLM inference



(a) and (b) analyze the latency in single-request and multi-request scenarios, respectively. (c) shows the GPU utilization of various computations during the generation phase on an A100 GPU.

Trade-off between Bandwidth and Capacity

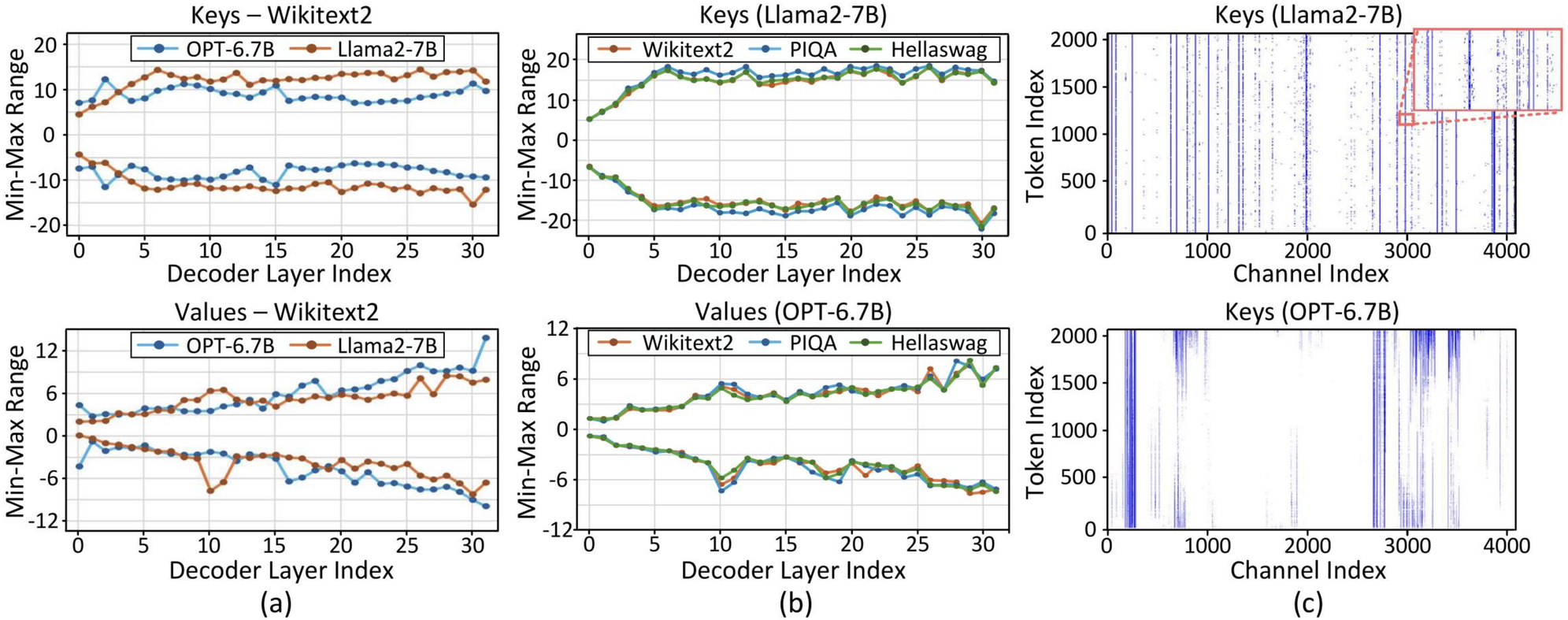




Left is the throughput under different batch sizes across different types of memory.

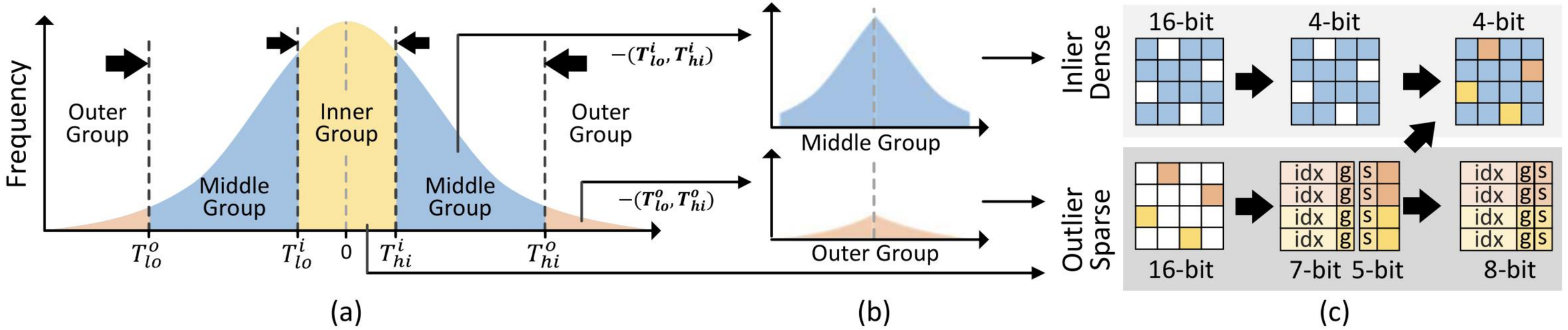
Right analyzes the effectiveness of KV cache quantization.

Characteristics of the data in KV cache



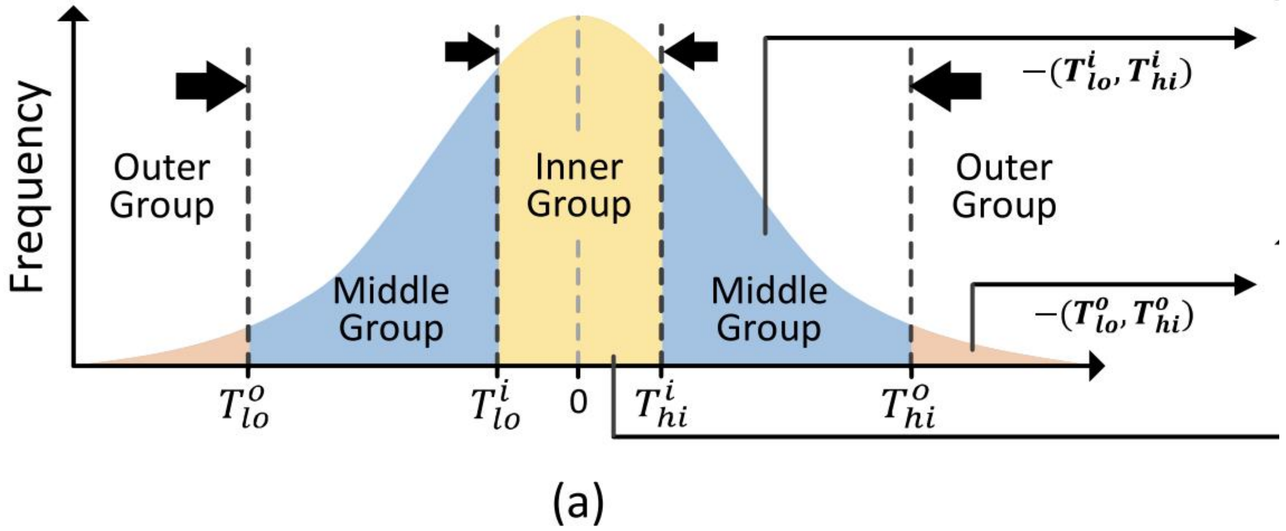
Line charts (a) and (b) show that the data distribution of the KV cache varies across different layers of different models, while remaining consistent across different inputs. Chart (c), after applying top-k selection, illustrates that the distribution within the same channel is non-uniform. Therefore, token-level quantization is necessary.

Algorithm



The algorithm decomposes the data distribution into three groups, then applies shift operations to reduce quantization loss, and finally combines dense and sparse encoding.

Offline outlier threshold profiling



(a)

$$G_o = \{x \mid x < T^o_{lo} \text{ or } T^o_{hi} < x\},$$

$$G_m = \{x \mid T^o_{lo} \le x < T^i_{lo} \text{ or } T^i_{hi} < x \le T^o_{hi}\},$$

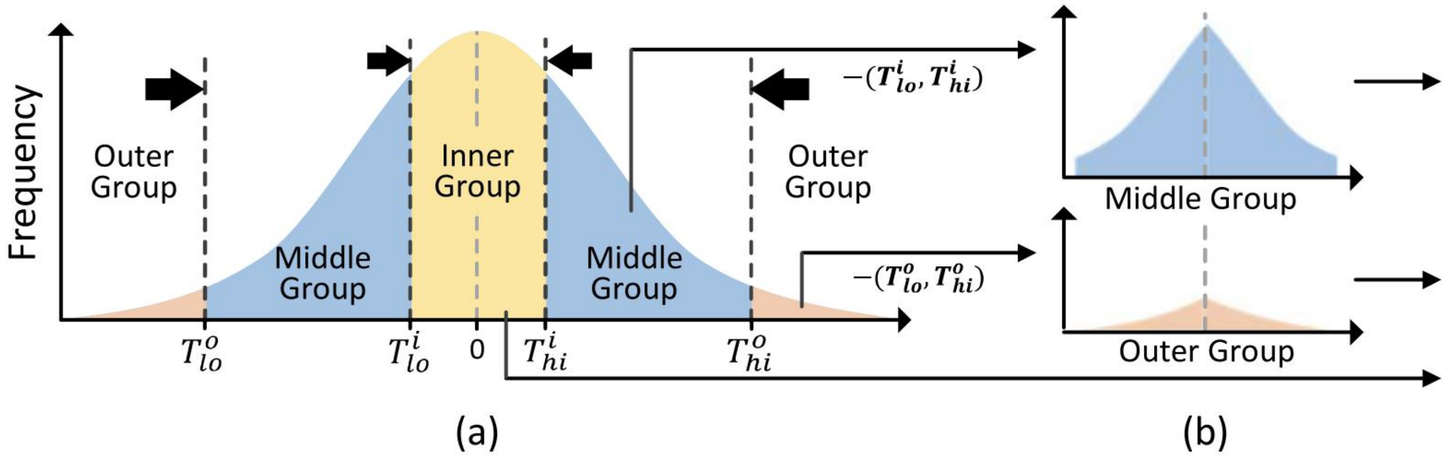
$$G_i = \{x \mid T^i_{lo} \le x \le T^i_{hi}\}$$

$$\sigma = \frac{2^m - 1}{\text{Max} - \text{Min}},$$

$$Q(x) = \text{round}((x - \text{Min}) \times \sigma).$$

Since the data distribution remains consistent across different inputs, the algorithm predefines thresholds for each model to classify the data. Within each category, uniform quantization is applied.

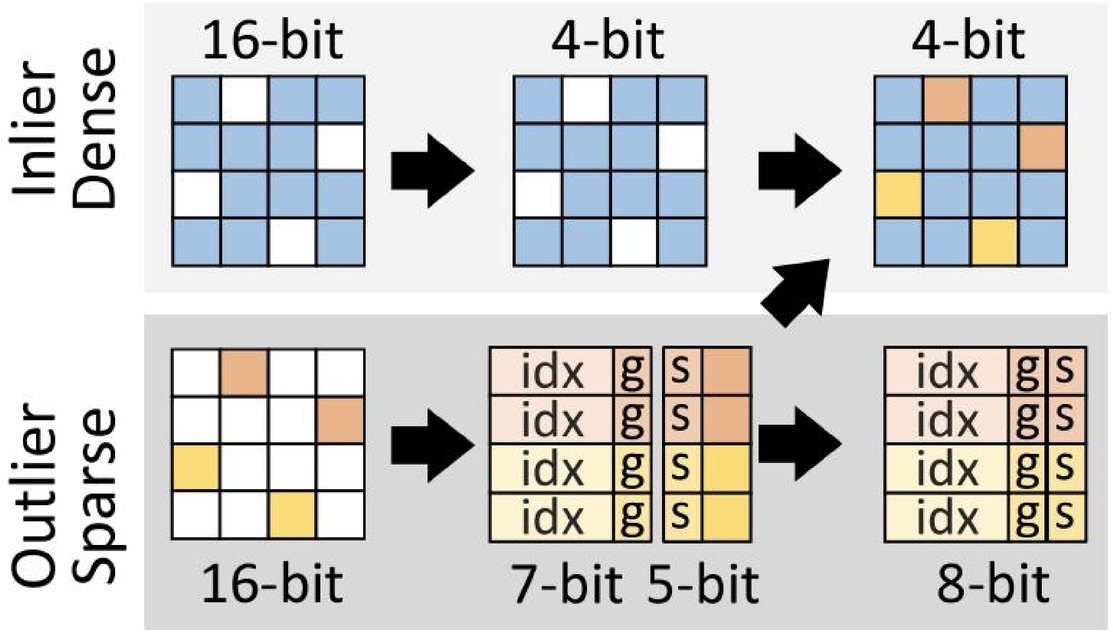
Group-shift algorithm



$$Q_o(x) = \begin{cases} Q(x - T^o_{hi}) & x \in G_o \text{ and } x > T^o_{hi} \\ Q(x - T^o_{lo}) & x \in G_o \text{ and } x < T^o_{lo} \\ Q(x - T^i_{hi}) & x \in G_m \text{ and } x > T^i_{hi} \\ Q(x - T^i_{lo}) & x \in G_m \text{ and } x < T^i_{lo} \\ Q(x) & x \in G_i \end{cases},$$

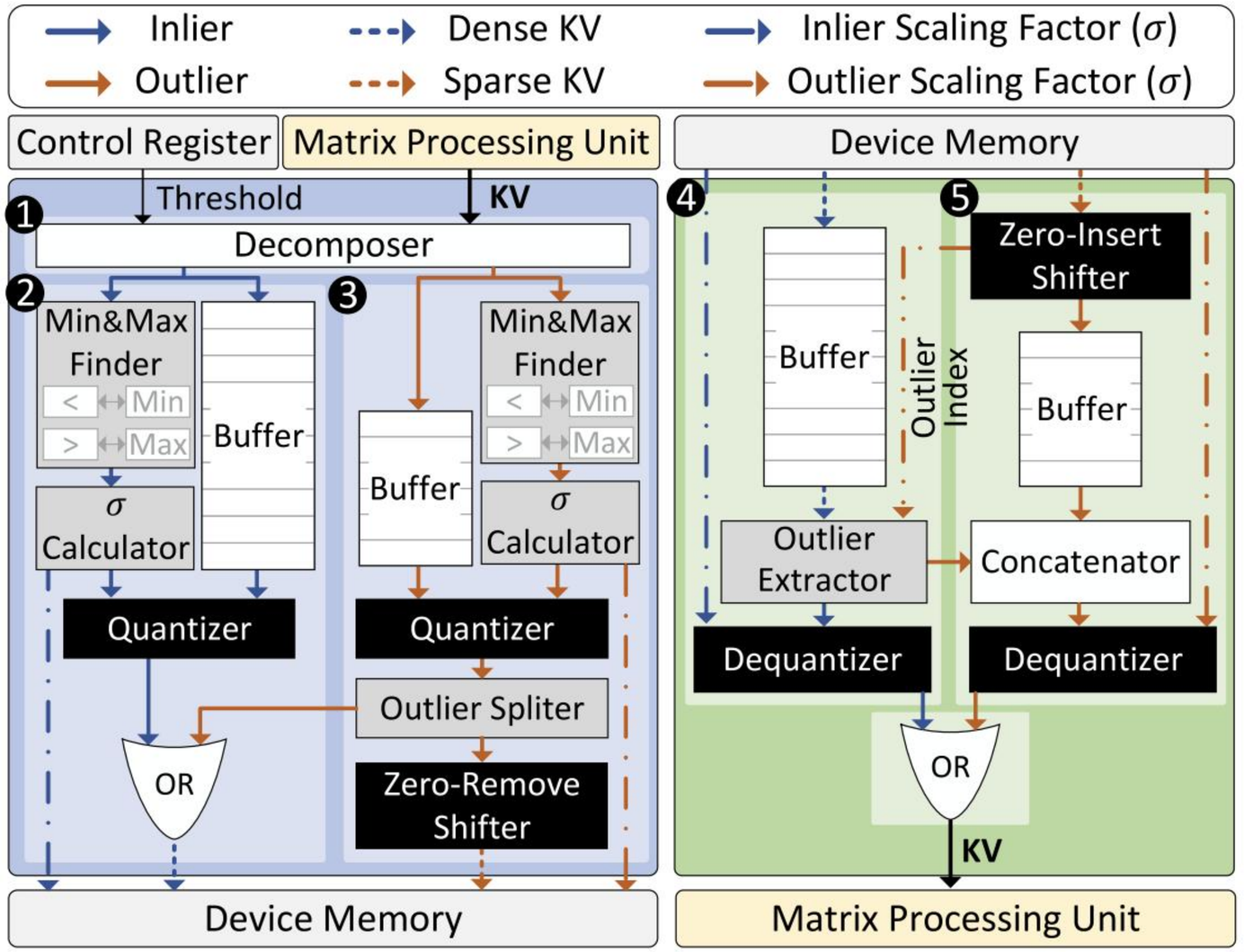
As mixed-precision methods are not hardware-friendly, the algorithm adopts a shift-based approach to handle outliers, thereby reducing their numerical range and subsequently minimizing quantization error. Oaken quantizes middle group into 4-bit, inner and outer groups into 5-bit.

Fused Dense-and-Sparse Encoding



Algorithm uses the Coordinate List (COO) format to store sparse matrix information. In the dense matrix, corresponding positions are zero. Standard COO uses 12 bits: 6 for position, 1 for group, and 5 for data. Fused Dense-and-Sparse Encoding shifts 4 data bits into the dense matrix, keeping only 1 sign bit in COO, reducing it to 8 bits.

Hardware Design



(a) Quantization Engine (b) Dequantization Engine

Evaluation results

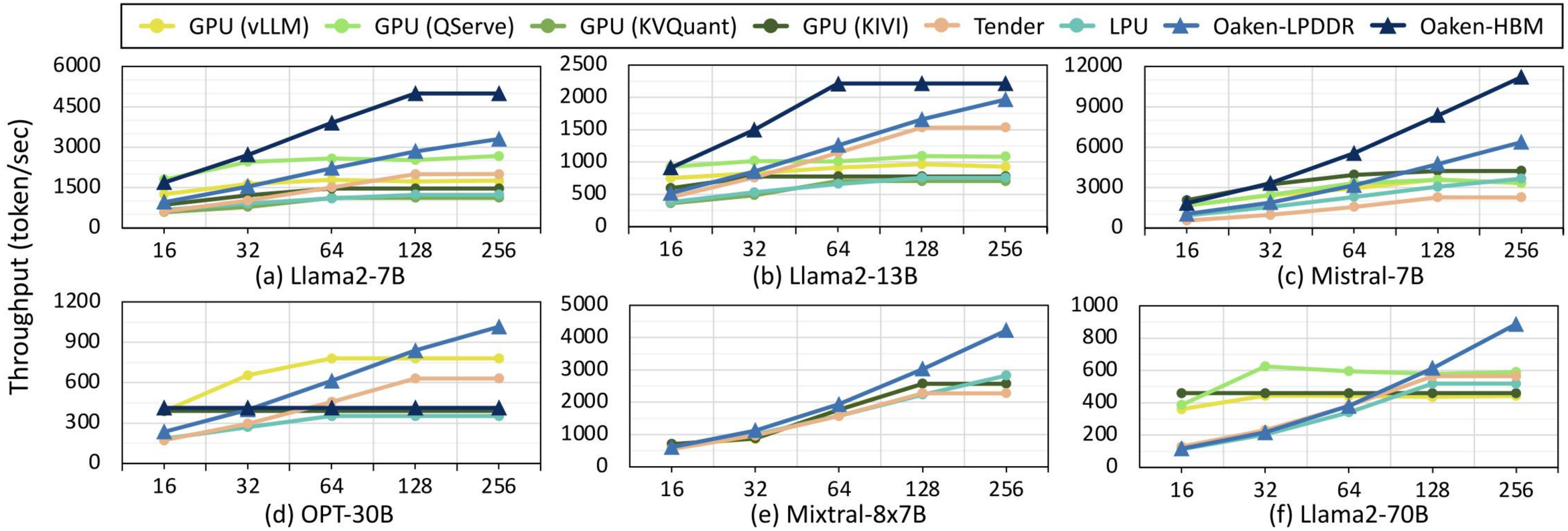


Figure 11: Throughput results of GPU baselines, LPU, Tender, and Oaken equipped with LPDDR and HBM across six LLMs. We sweep the batch size from 16 to 256. The input and output sequence lengths are set to 1K:1K.

Evaluation results

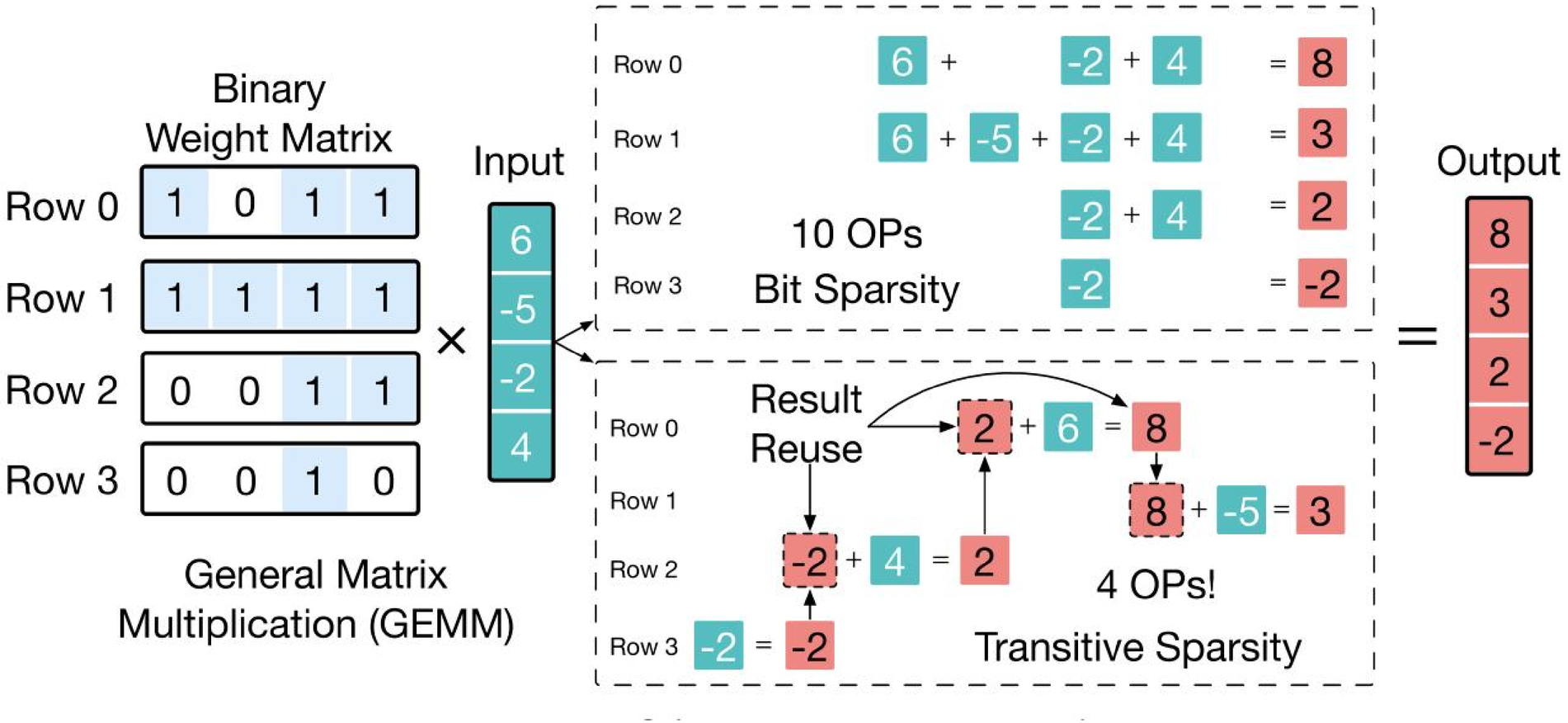
Table 2: Perplexity results on Wikitext2, and zero-shot accuracy results on PIQA, Winogrande, and Hellaswag datasets with effective bitwidth of each quantization technique.

Model	Llama2			OPT			Mistral	Mixtral	Llama2			Llama2			Llama2			Llama2		
	7B	13B	70B	6.7B	13B	30B	7B	8x7B	7B	13B	70B	7B	13B	70B	7B	13B	70B	7B	13B	70B
Task	Wikitext2								PIQA			Winogrande			Hellaswag			-		
Metric	Perplexity (↓)								Accuracy (%)			Accuracy (%)			Accuracy (%)			Effective Bitwidth		
Original	**5.47**	**4.88**	**3.32**	**10.86**	**10.13**	**9.56**	**5.25**	**3.84**	**79.05**	**80.52**	**82.70**	**69.13**	**72.80**	**80.20**	**75.98**	**79.38**	**83.82**	**16.00**	**16.00**	**16.00**
KVQuant	5.49	4.94	3.33	10.88	10.14	9.58	5.33	3.87	78.35	79.33	82.21	67.80	71.74	77.98	75.82	79.25	83.70	4.82	4.81	5.01
KIVI	5.50	4.90	3.33	10.88	10.16	9.58	5.34	3.84	78.07	79.05	78.07	67.84	70.96	76.81	75.57	78.97	83.47	4.99	4.99	4.99
Tender	6.42	5.74	4.25	11.80	11.05	10.44	5.54	NaN	74.27	76.12	77.91	62.90	65.69	74.59	73.89	77.16	75.04	4.07	4.07	4.10
Atom	5.62	4.98	3.37	11.01	10.22	9.64	5.42	4.05	76.17	76.99	81.34	66.46	67.09	75.77	72.22	76.21	80.52	4.25	4.25	4.63
QServe	5.67	5.12	3.36	10.95	10.28	9.62	5.42	4.03	77.37	77.48	81.77	65.29	66.80	76.09	74.41	76.69	83.24	4.25	4.25	4.25
Oaken	**5.53**	**4.93**	**3.34**	**10.88**	**10.16**	**9.58**	**5.35**	**3.90**	**78.29**	**79.71**	**82.59**	**67.64**	**70.56**	**76.64**	**73.72**	**78.24**	**83.50**	**4.82**	**4.82**	**4.89**

Transitive Array

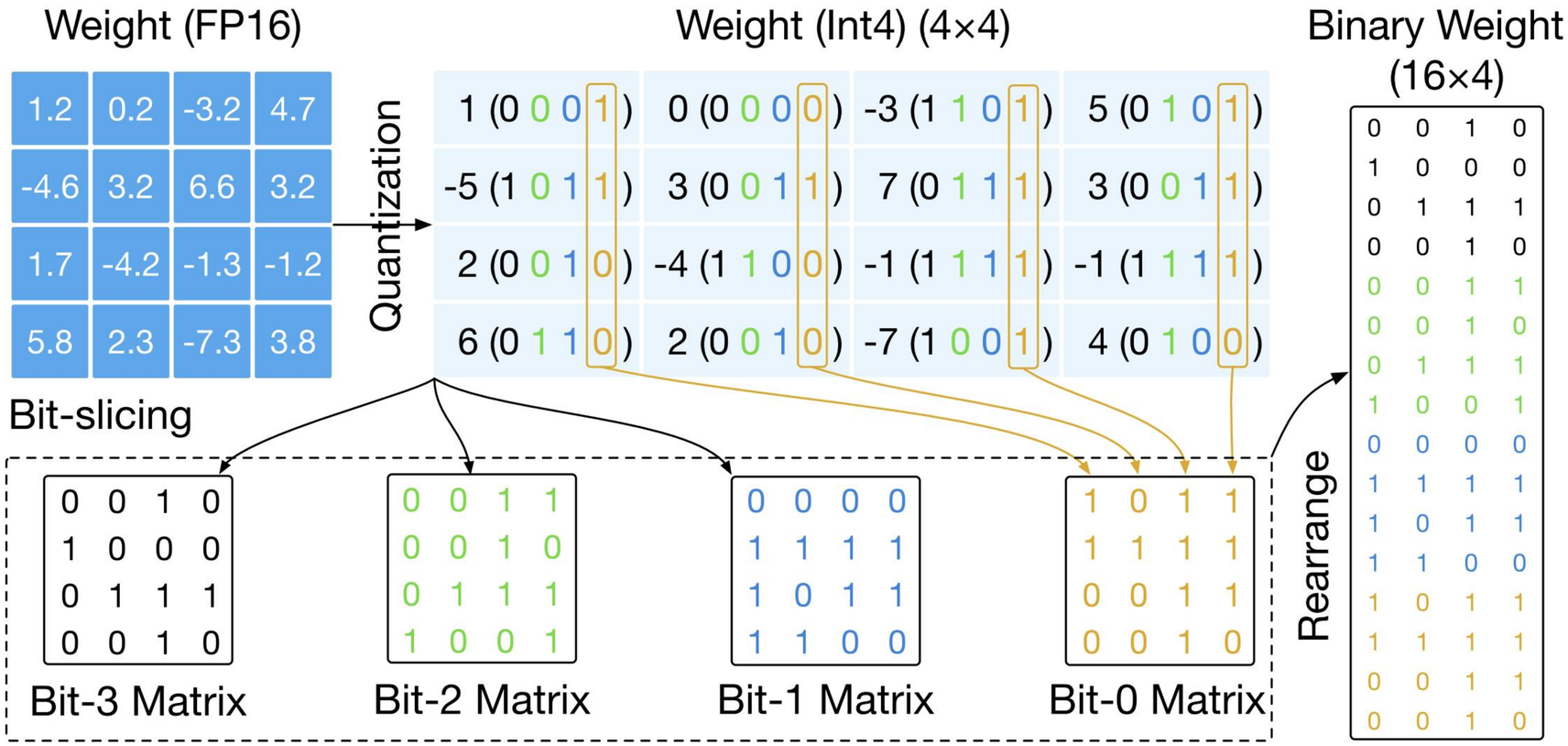
An Efficient GEMM Accelerator with Result Reuse

An Example



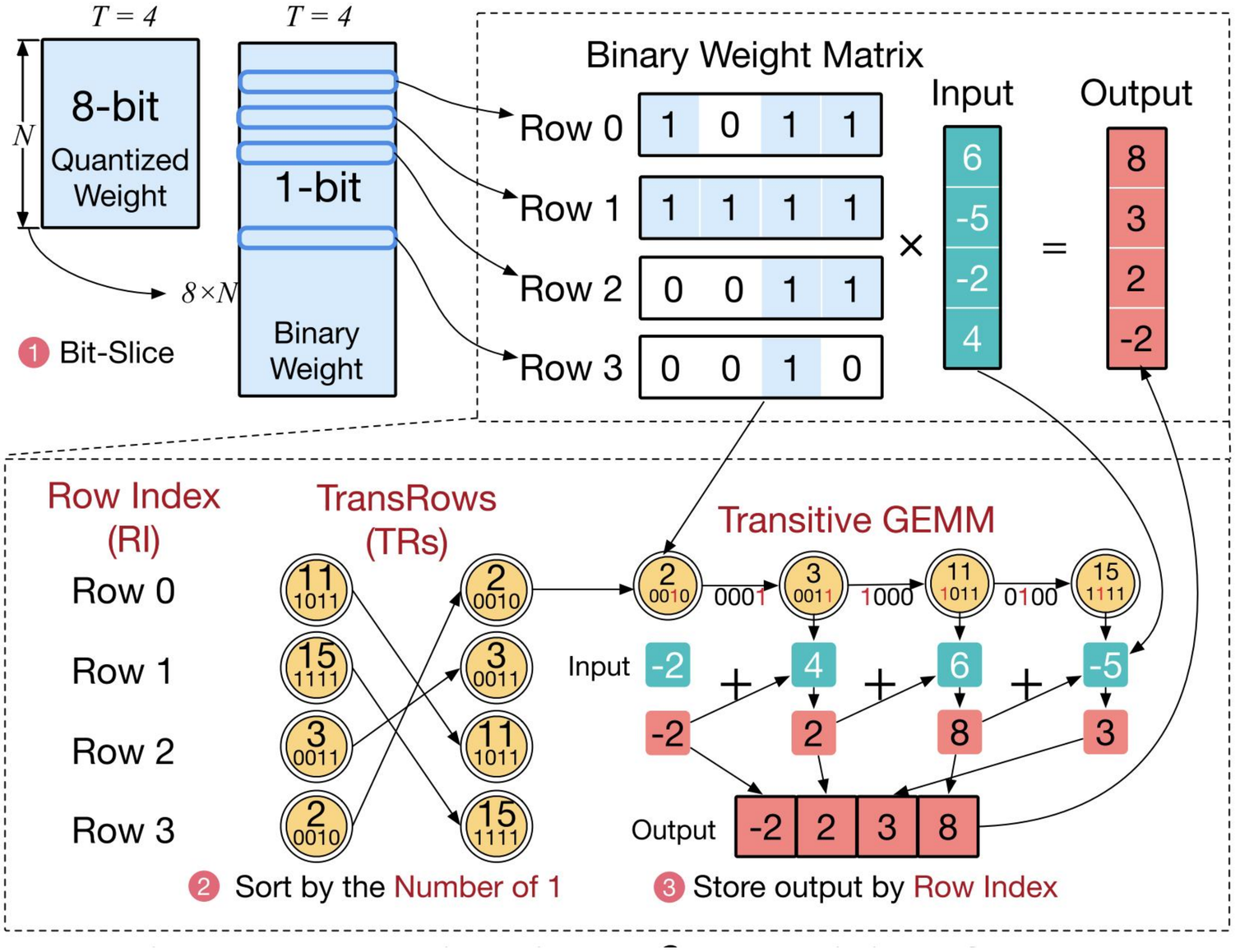
Reusing intermediate results across different rows can significantly reduce the compu

Bit Slicing

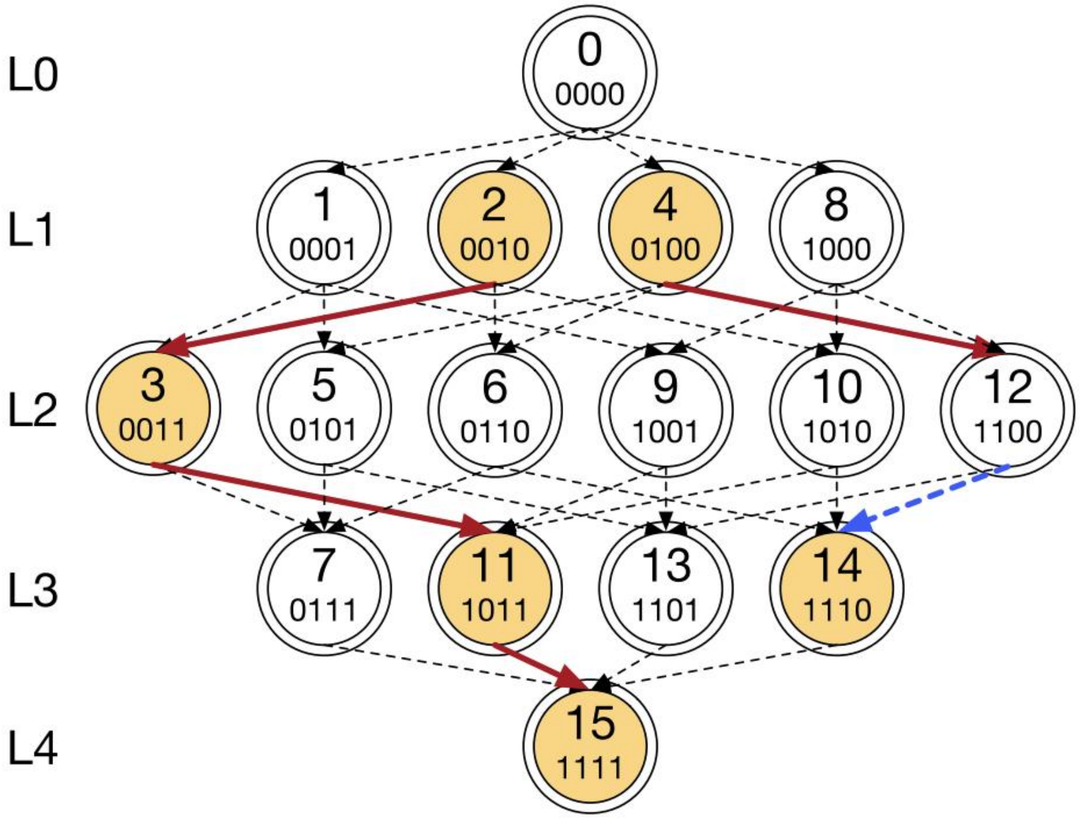


In general, for an S -bit quantized matrix of shape $(N \times K)$, we can reorganize it into

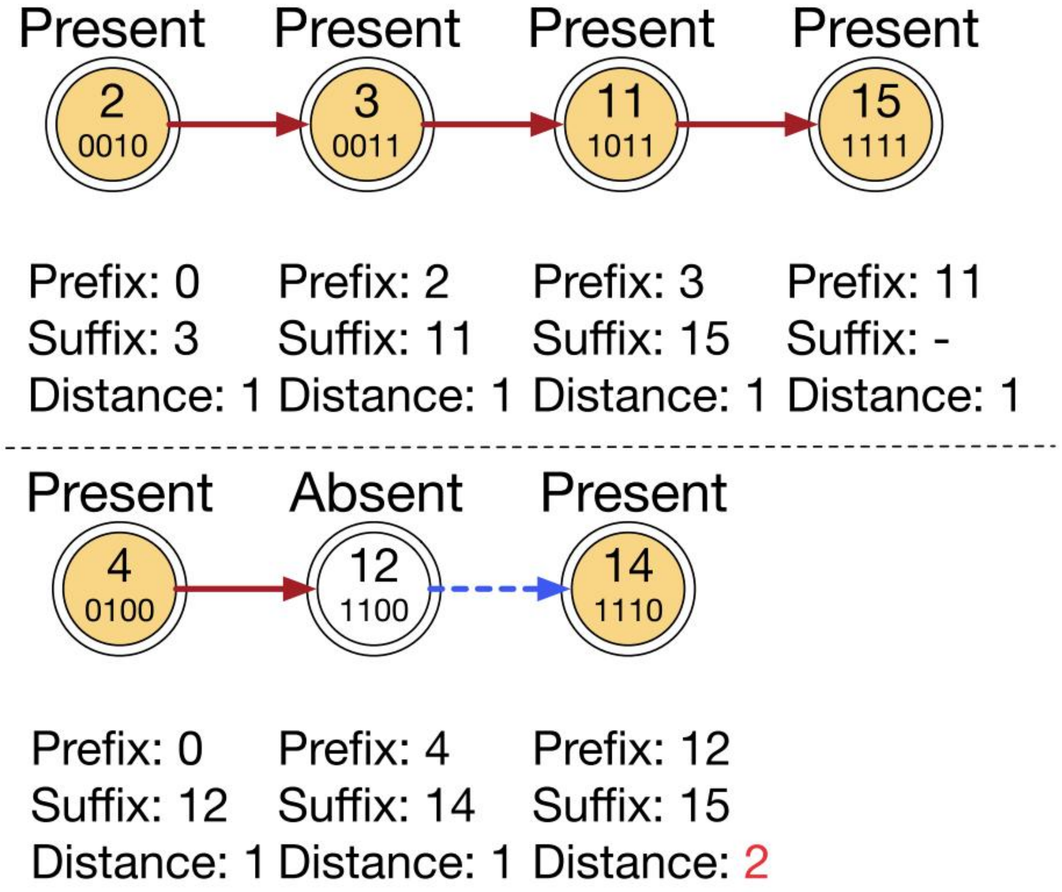
Motivation of Transitive of GEMM



Hasse Graph

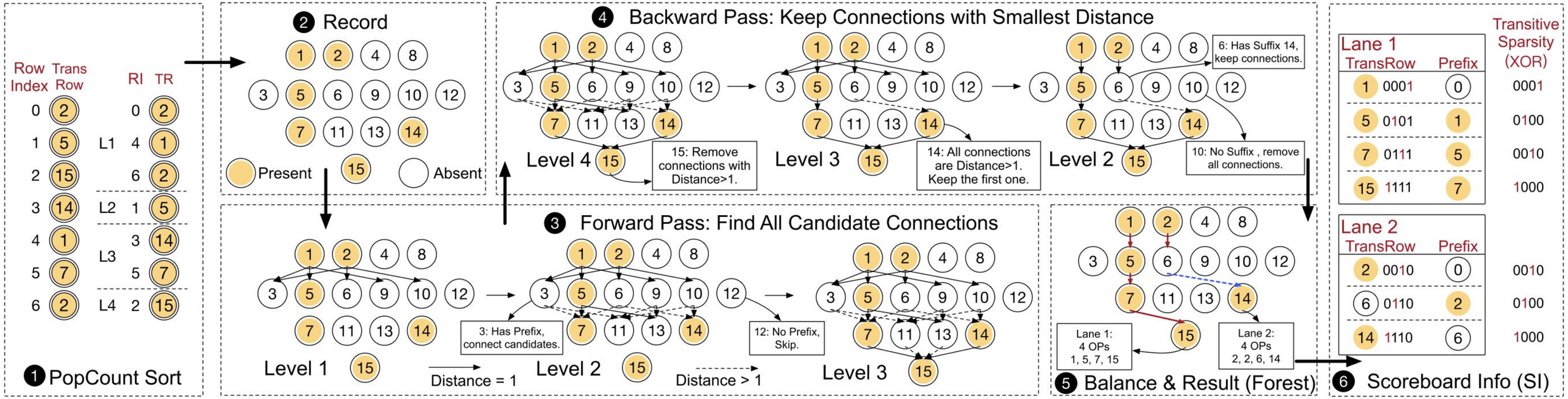


(a) Hasse Graph



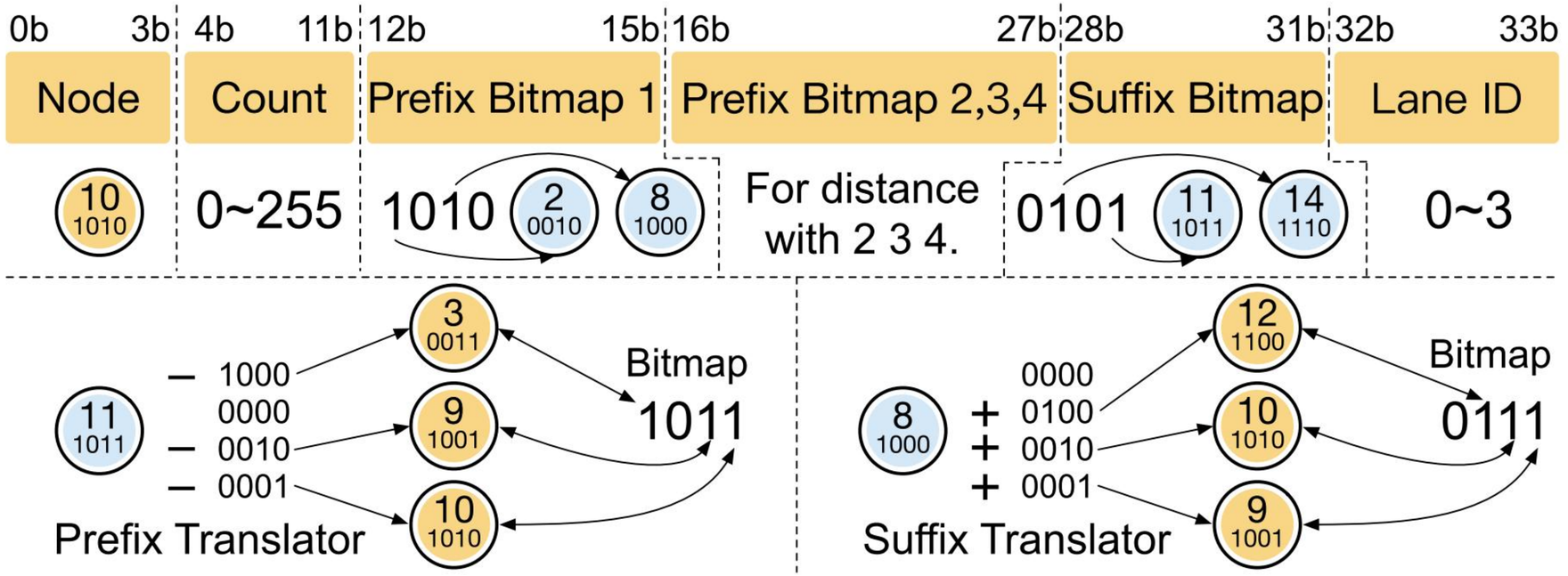
(b) Prefix, Suffix, and Distance

Algorithm



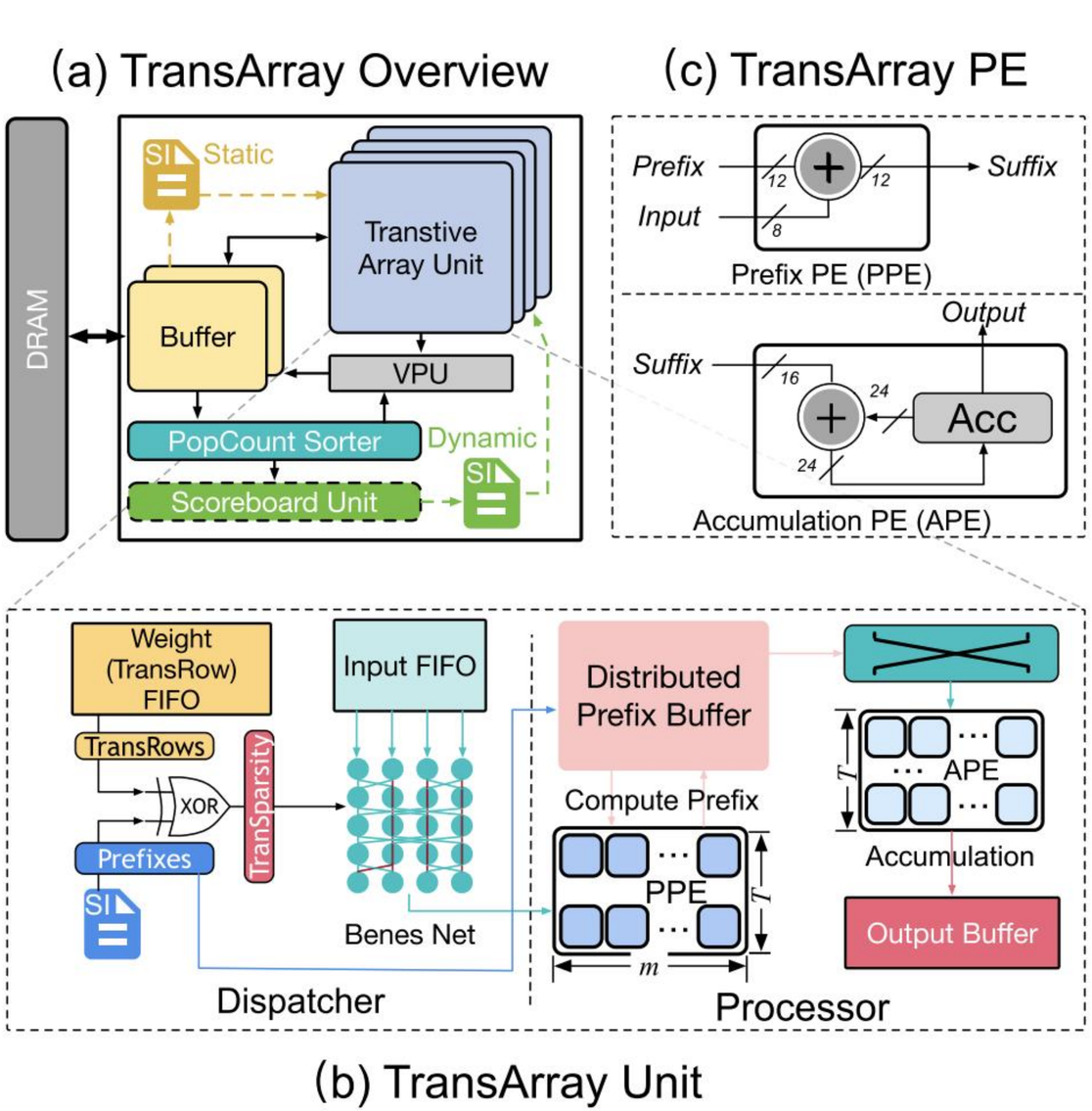
Since GEMM in LLM is computed in a tiled manner, the most suitable approach is to compute the scoreboard online.

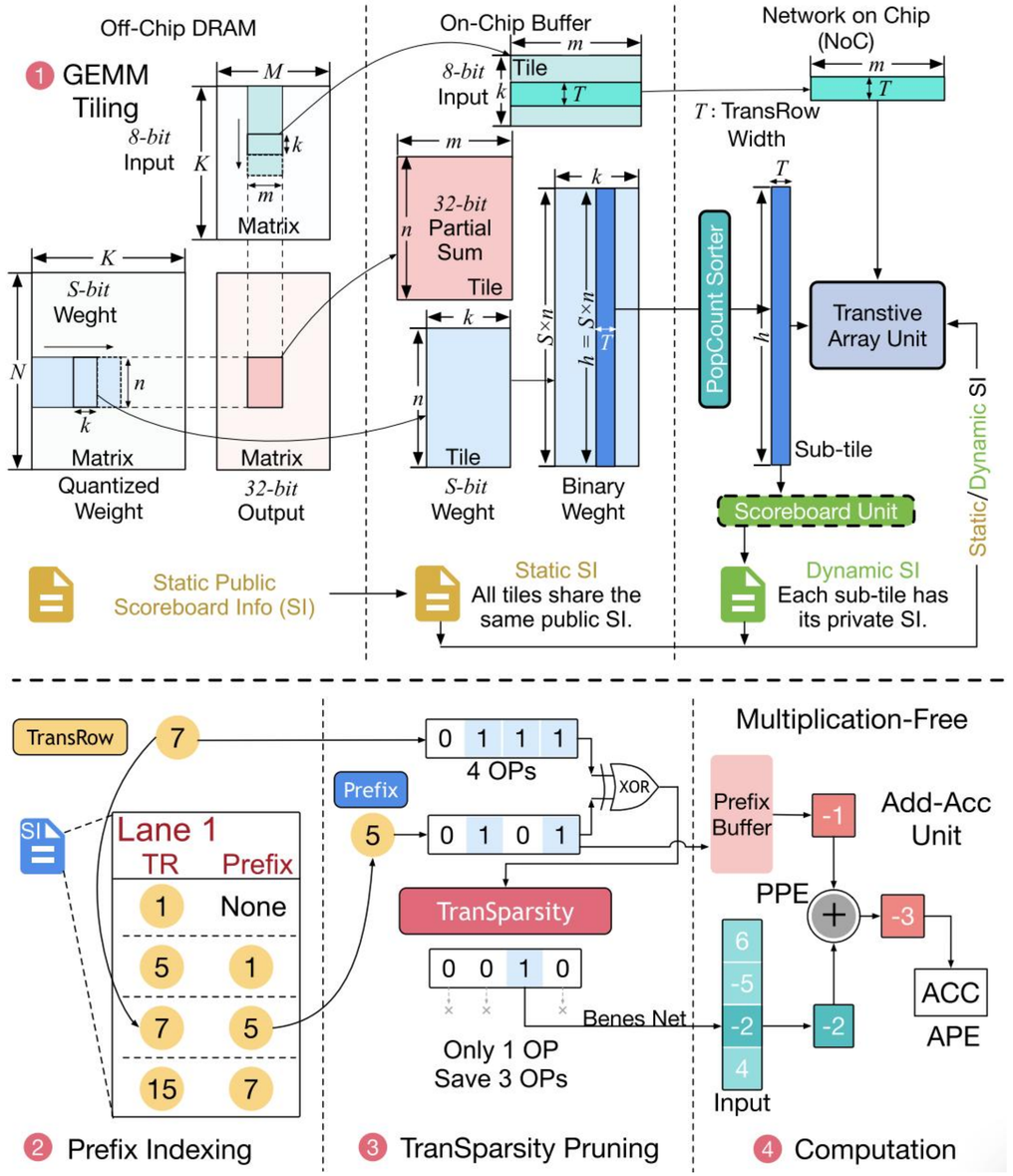
Bit field diagram of Score board



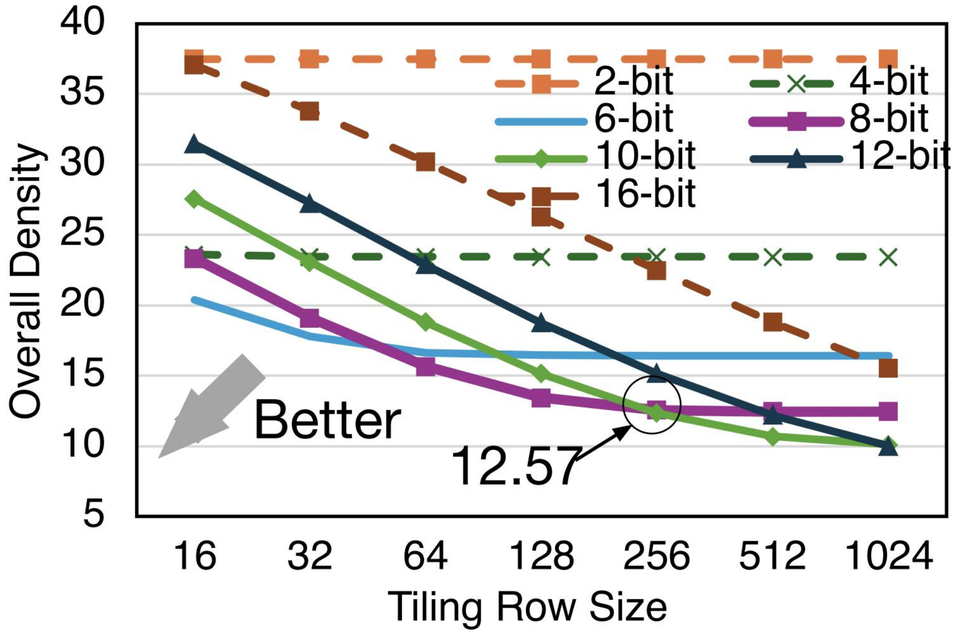
The prefix translator and suffix translator employ bit-reversal operations to efficiently construct forward and backward nodes.

Hardware Design





Evaluation Results

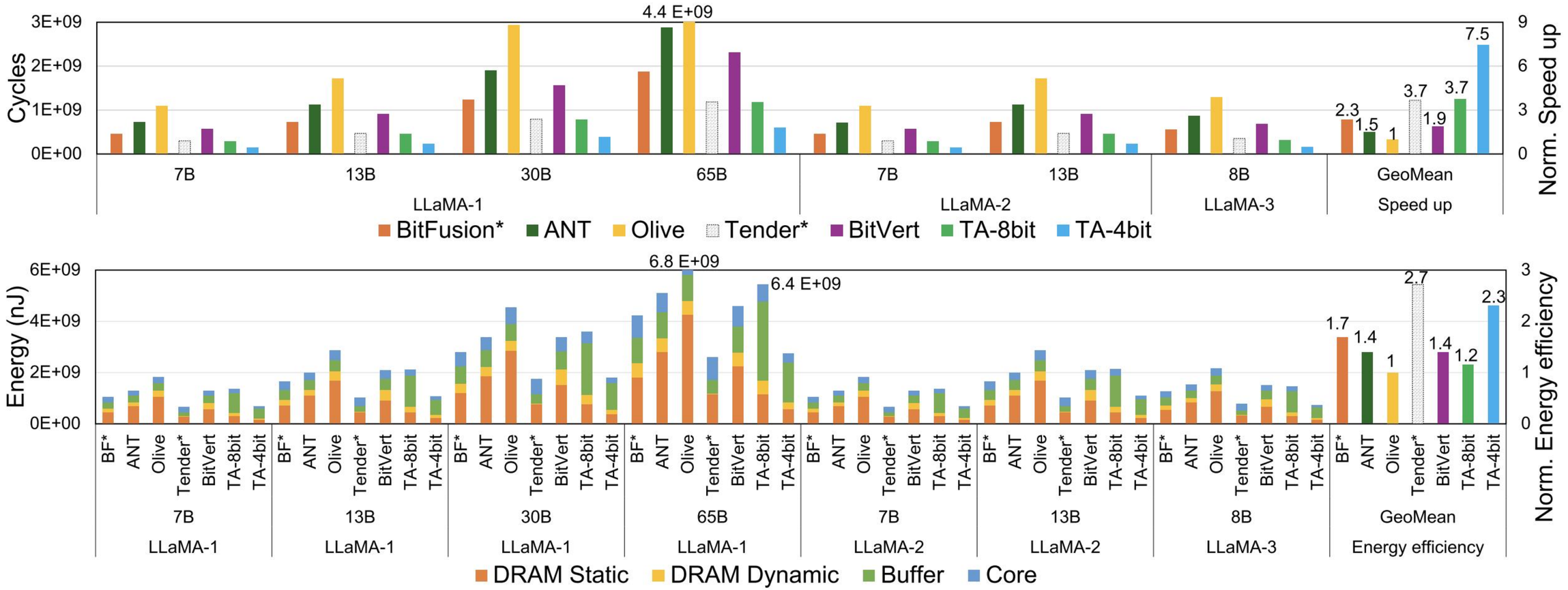


Sparsity of different tile size

Arch.	Computation Core			Buffer
	Component	Array	Area (mm^2)	
TransArray (6 Units)	PPE ($50.3\mu m^2$)	$6 \times (8 \times 32)$	**0.443**	480KB
	APE ($101.7\mu m^2$)	$6 \times (8 \times 32)$		
	NoC ($19520\mu m^2$)	$6 \times (1)$		
	Socreboard ($92507\mu m^2$)	1		
BitFusion	8-bit PE ($548\mu m^2$)	28×32	0.491	512KB
ANT	4-bit PE ($210\mu m^2$)	36×64	0.484	
Olive	4-bit PE ($319\mu m^2$)	32×48	0.489	
BitVert	8-bit PE ($985\mu m^2$)	16×30	0.473	
Tender	4-bit PE ($329\mu m^2$)	30×48	0.474	608KB

Area size

Evaluation Results



Runtime and energy consumption on FC layers of LLaMA models.

Thank you!