Mustafar: Promoting Unstructured Sparsity for KV Cache Pruning in LLM Inference

Donghyeon Joo[1], Helya Hosseini[1], Ramyad Hadidi[2], Bahar Asgari[1]

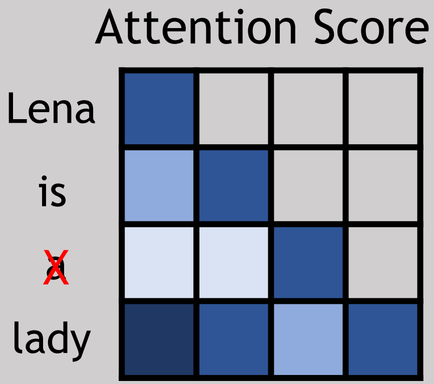
[1]University of Maryland, College Park

[2]d-Matrix

KV cache size scales with long-context

Various KV compression techniques are used:

Token Eviction

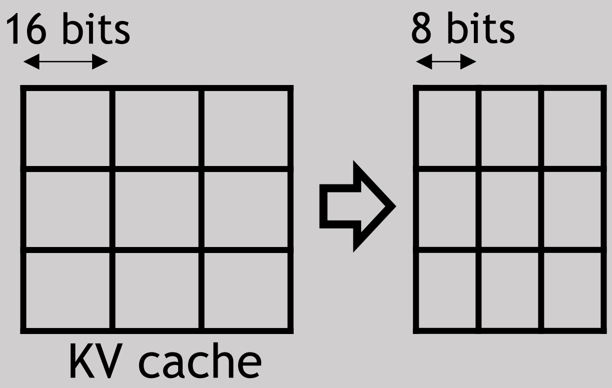


Evict KV cache of Less-critical Tokens

H2O [NeurIPS 2023]

HeadKV [ICLR 2025]

Quantization

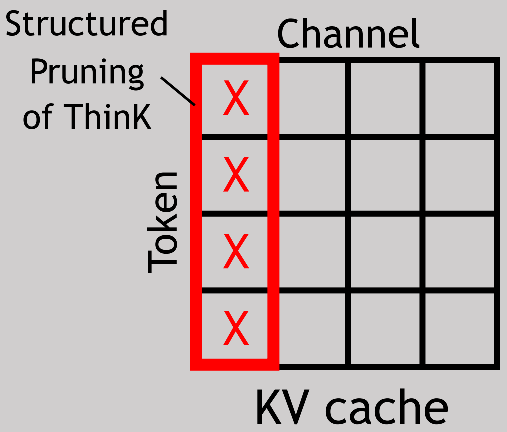


Reduce KV bit-width

KIVI [ICML 2024]

ZipCache [NeurIPS 2024]

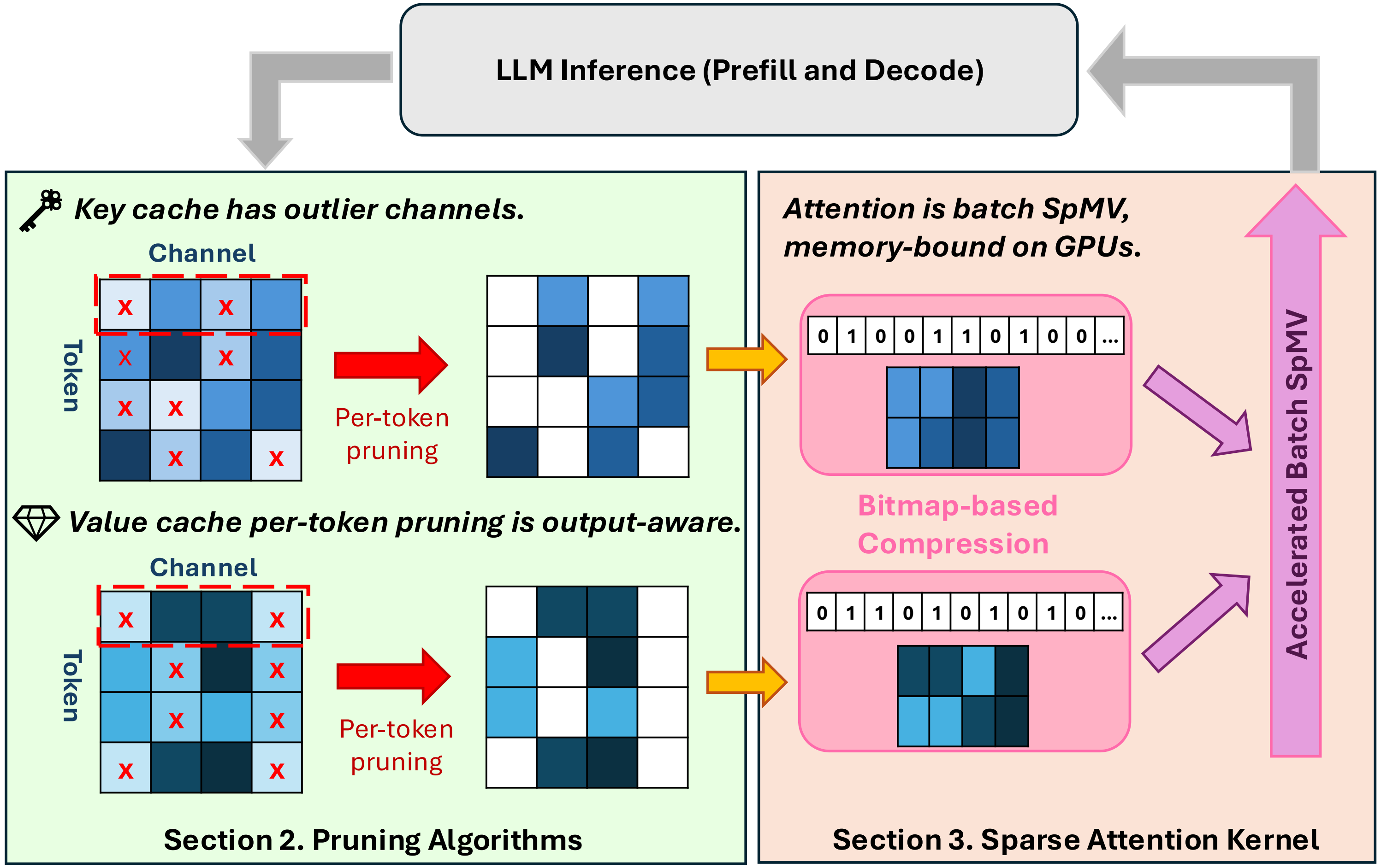
Pruning



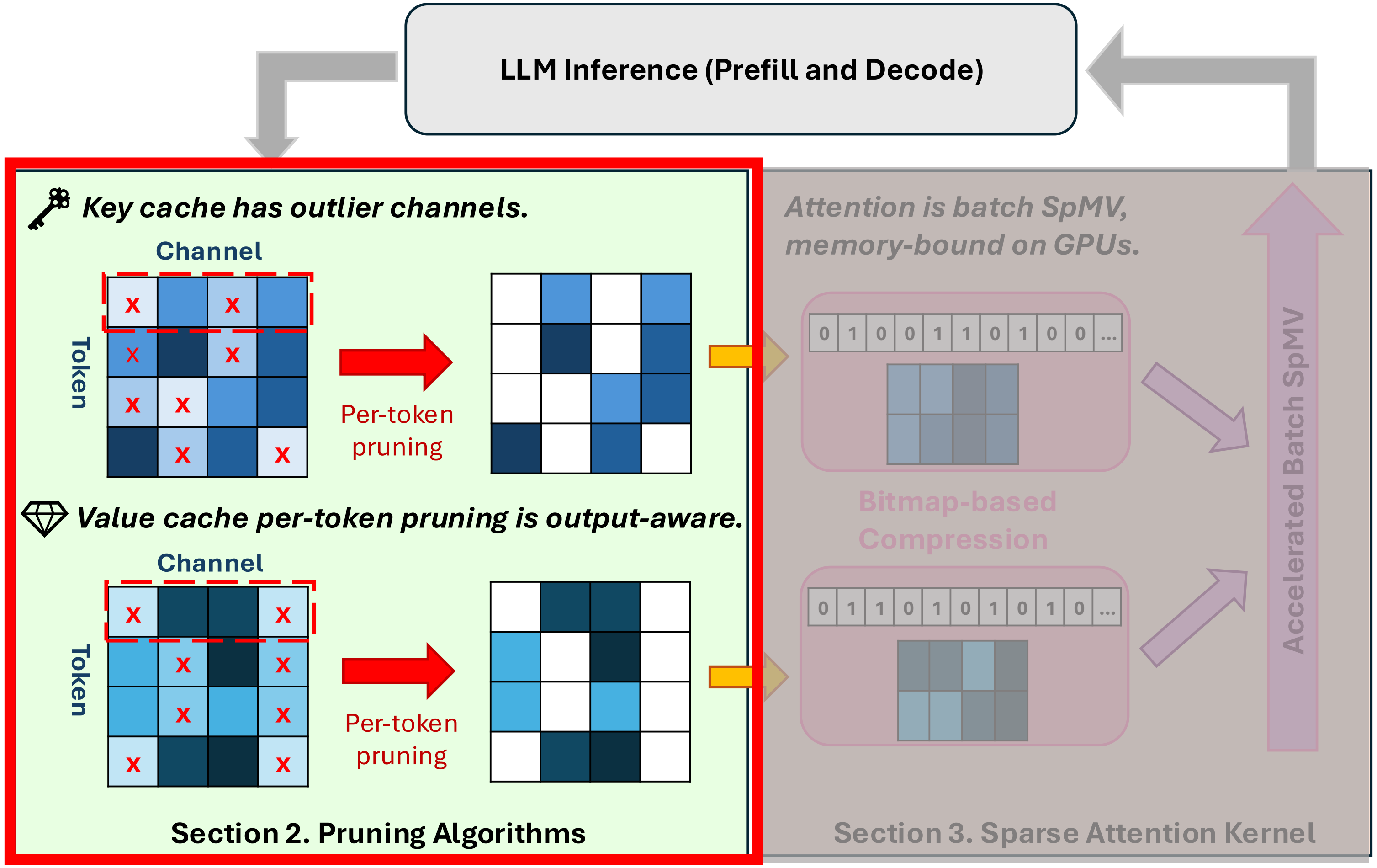
Reduce KV Matrices

ThinK [ICLR 2025]

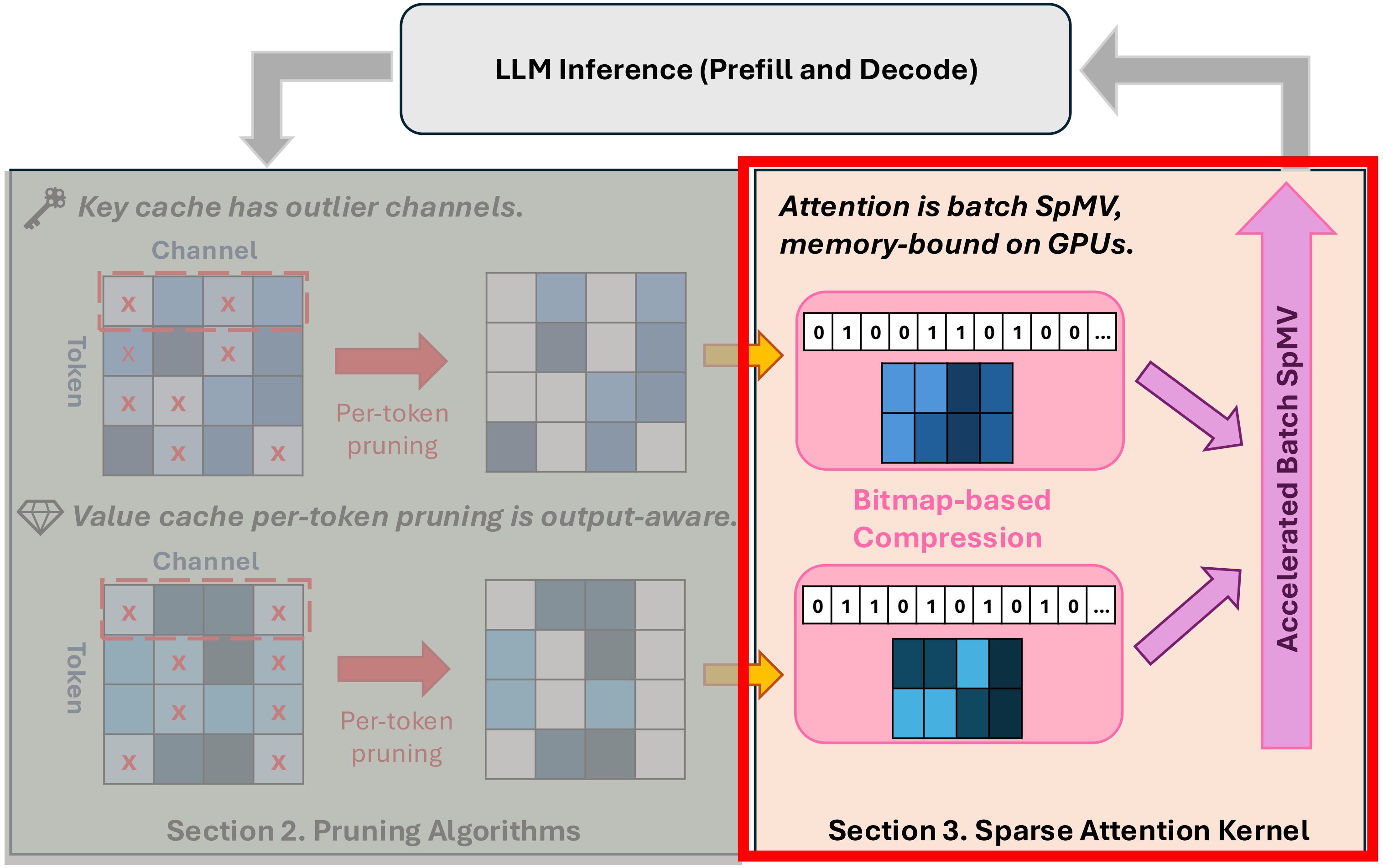
Mustafar Overview



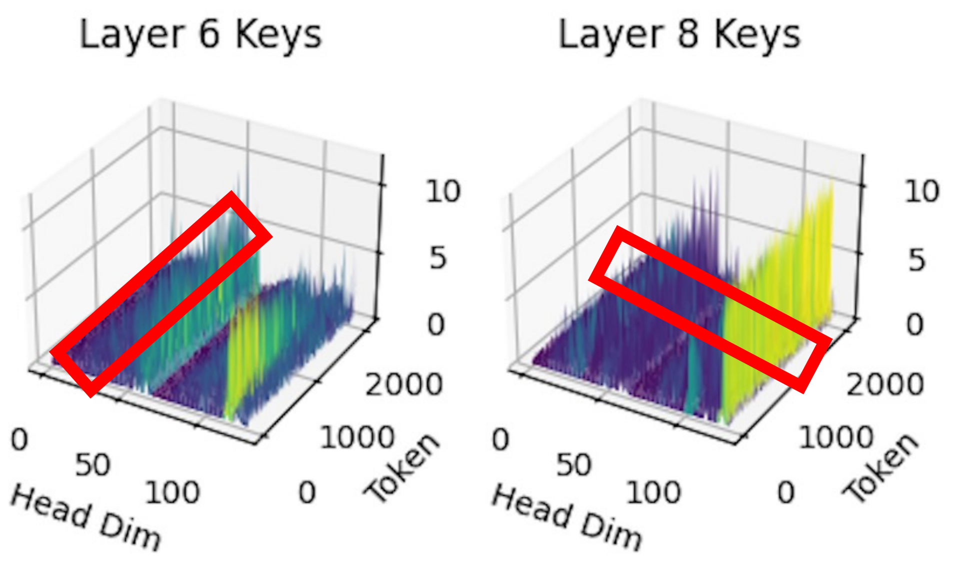
KV Cache Unstructured Pruning



Unstructured Sparse Attention Kernel



Key Cache Observation



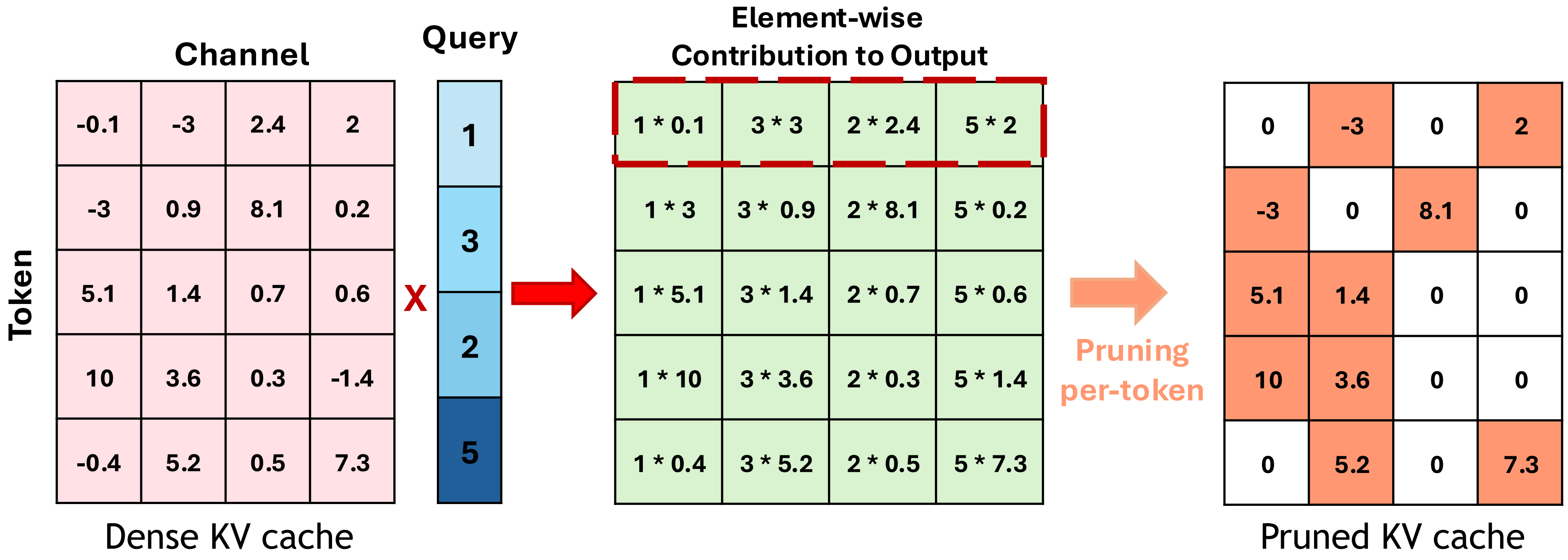
Key Cache Magnitude Distribution

- Key cache shows distinct channel-wise outliers.
- ThinK applied channel-wise structured pruning.
  - But can unstructured sparsity do better?
- Pruning direction should be per-token.

Visualization credit to
KIVI (Liu et al. ICML 2024)

Key Cache Pruning

- Pruning Strategy #1: Magnitude-based pruning
- Pruning Strategy #2: Output-aware pruning



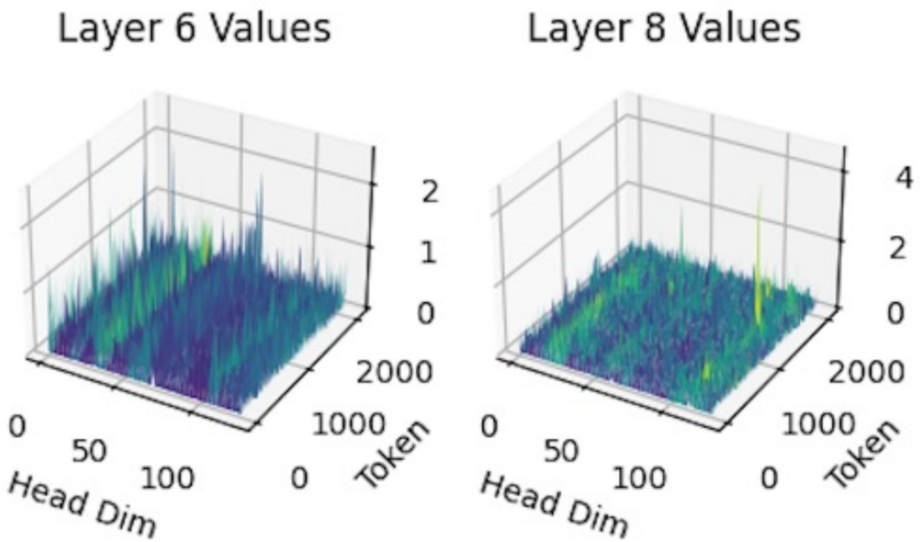
Per-Token Output-aware Pruning

Llama-3-8B-Instruct Accuracy on LongBench

- ThinK significantly degrades accuracy at 70% sparsity.
- Both unstructured pruning preserves accuracy.
  - Magnitude-based pruning is selected for runtime efficiency.

Task	Dense	$K_s = 0.5$			$K_s = 0.7$		
		ThinK (Structured)	Unstructured Output-aware	Unstructured Magnitude	ThinK (Structured)	Unstructured Output-aware	Unstructured Magnitude
Average	43.19	38.53	**43.23**	42.84	26.55	**42.13**	41.55
SingleDoc QA	36.66	35.61	36.57	**36.90**	25.26	**35.78**	35.53
MultiDoc QA	36.09	34.99	**35.92**	35.77	29.75	**35.55**	35.40
Summarization	26.75	24.96	**26.87**	26.45	17.70	25.16	**25.18**
Few-shot	68.96	66.54	**68.82**	68.75	44.88	67.22	**67.84**
Synthetic	37.25	35.50	**37.00**	36.75	16.86	**35.25**	35.00
Code	55.58	29.56	**56.61**	54.14	19.15	**56.19**	51.47

Value Cache Observation

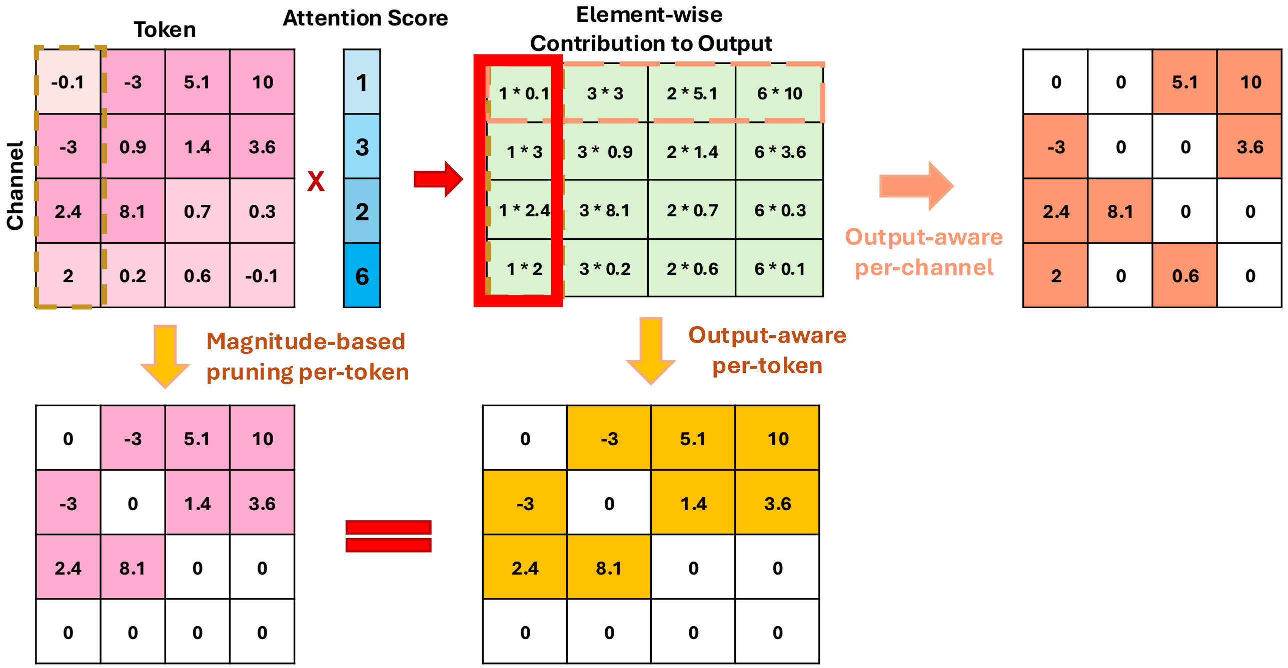


Value Cache Magnitude Distribution

- Value cache exhibits more uniform distribution.
- ThinK reported to be ineffective.
- Both pruning directions must be explored.

Value Cache Pruning

- Pruning Strategy #1: Per-channel magnitude-based pruning
- Pruning Strategy #2: Per-channel output-aware pruning
- Pruning Strategy #3: Per-token magnitude-based pruning, is already output-aware!



Llama-3-8B-Instruct Accuracy on LongBench

- ThinK significantly degrades accuracy at 70% sparsity.
- Per-token magnitude-pruning is both effective and efficient.
- Per-token pruning is jointly applicable with token eviction and quantization.

Task	Dense	$V_s = 0.5$				$V_s = 0.7$			
		ThinK (Structured)	Magnitude (Per-channel)	Output-aware (Per-channel)	Magnitude (Per-token)	ThinK (Structured)	Magnitude (Per-channel)	Output-aware (Per-channel)	Magnitude (Per-token)
Average	43.19	38.45	42.50	42.84	**43.04**	30.60	41.69	42.67	**42.78**
SingleDoc QA	36.66	34.92	36.56	36.24	**36.75**	25.05	36.11	36.05	**36.96**
MultiDoc QA	36.09	34.74	35.45	36.07	**36.22**	23.90	35.11	**36.20**	35.82
Summarization	26.75	23.31	24.74	25.79	**26.34**	20.41	22.72	24.75	**25.19**
Few-shot	68.96	67.18	67.66	68.65	**68.91**	60.16	67.39	**68.23**	68.08
Synthetic	37.25	35.43	**38.31**	37.00	36.25	29.63	**38.75**	37.25	35.50
Code	55.58	31.97	55.07	55.57	**55.77**	20.85	52.65	56.17	**57.62**

Bitmap-based Sparse Format

- Objective #1: Maximally compress unstructured sparse KV cache
- Compress unstructured sparse KV cache with a bitmap-based sparse format.

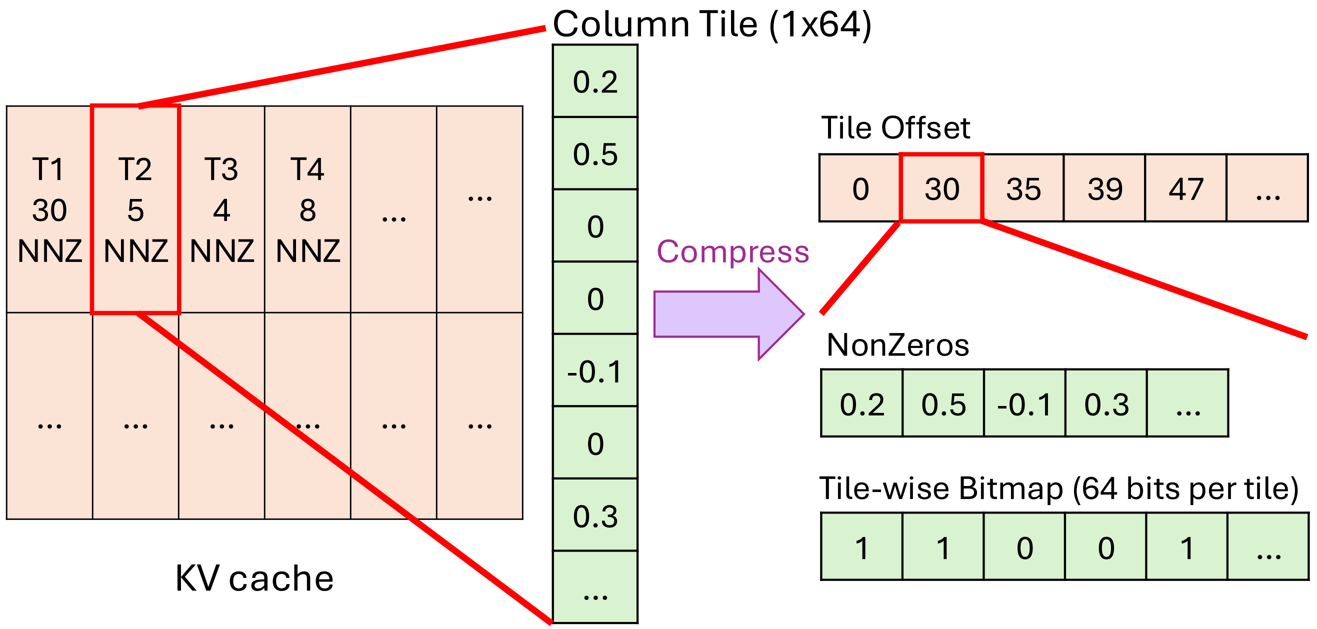


Figure credit to
Coruscant (Joo et al. MICRO 2025)

Load-as-compressed, Compute-as-dense Pipeline

- Objective #2: Accelerate memory-bound decode attention computation
- Load from GPU GMEM to SMEM in compressed form, compute as dense in TC.

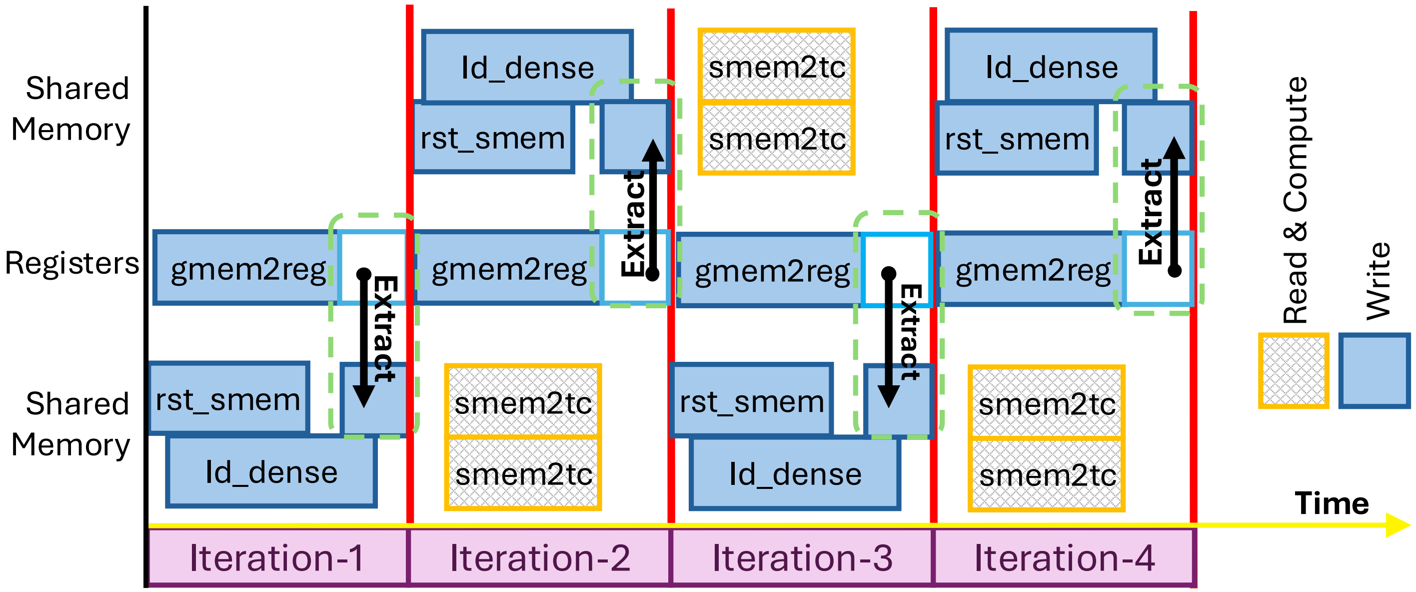
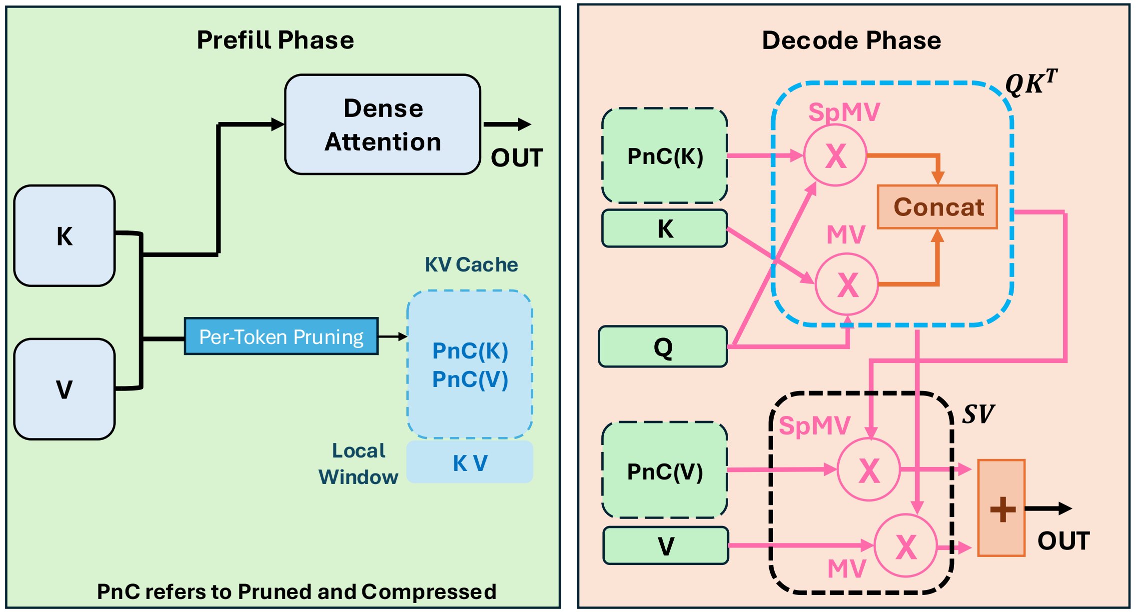


Figure credit to
Flash-LLM (Xia et al. VLDB 2023)

Mustafar Sparse Attention Kernel

- KV cache is pruned and compress on-the-fly.
- Decode attention is computed as a combination of sparse attention on compressed cache and dense attention on local dense cache.



*multi-head, softmax, and normalization are omitted for simplicity.

Evaluation Methodology

- **System:** NVIDIA RTX 6000 ADA GPU
- **Models:**
  - Llama-2 7B/13B, Llama-3/3.1-8B-Instruct, Mistral-7B-Instruct-v0.2
- **Key Metrics:**
  - **Accuracy:** LongBench and RULER
  - **Efficiency:** Compression ratio, kernel latency, token throughput, TTFT, decode speed

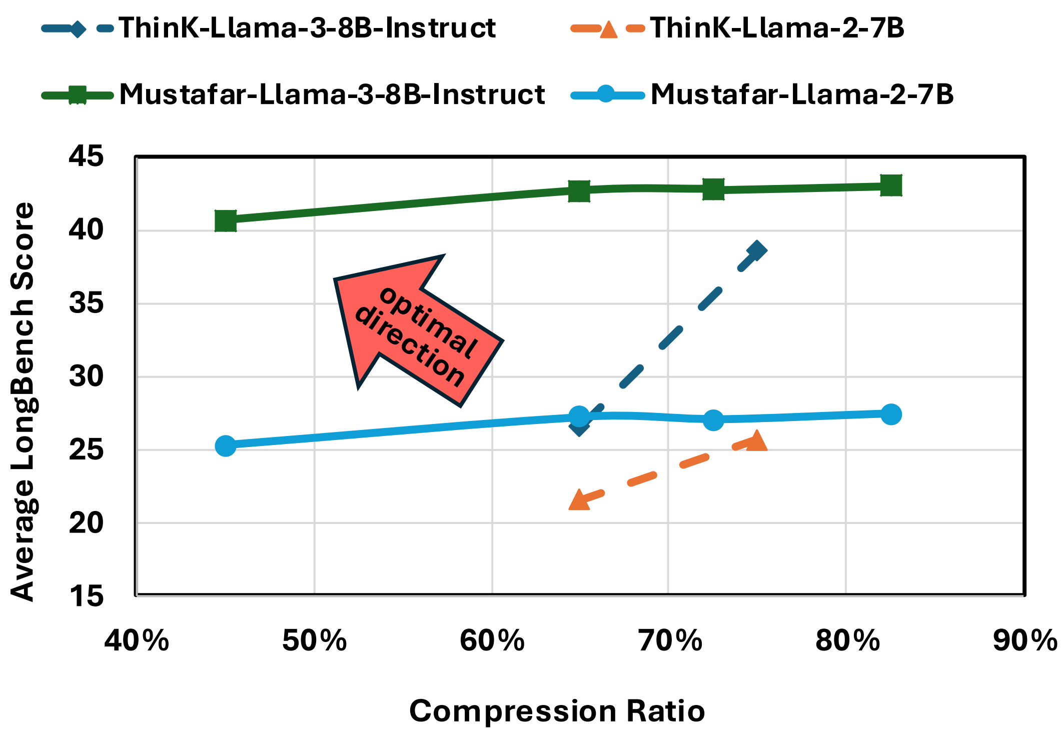
Evaluation: Accuracy

- Mustafar preserves accuracy even when both Key and Value caches are pruned.
- Constantly observed across all models tested.

KV Sparsity	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NtrvQA	Qasper	MF-en	HotpotQA	2WikiMQA	Musique	GovReport	QMSum	MultiNews	TREC	TrivialQA	SAMSum	PCount	PRe	Lcc	RBP	
Llama-3 8B Instruct																	
Dense	23.39	43.38	43.22	46.39	38.66	23.22	29.91	22.56	27.77	74.50	90.28	42.11	4.50	70.00	57.11	54.05	**43.19**
ThinK0.5	22.38	40.96	43.48	44.01	38.37	22.59	26.61	22.20	26.08	74.00	88.83	36.79	6.00	65.00	27.95	31.17	**38.53**
K0.5 V0.0	23.40	43.68	43.63	46.00	38.60	22.72	29.39	22.33	27.64	74.50	90.66	41.09	5.00	68.50	55.89	52.39	**42.84**
ThinK0.7	17.58	27.40	30.80	40.59	29.50	19.16	18.13	17.28	17.70	34.00	83.09	17.56	4.71	29.00	17.88	20.42	**26.55**
K0.7 V0.0	22.91	42.36	41.33	45.53	38.50	22.16	26.63	21.90	27.00	73.00	90.83	39.68	4.50	65.50	51.94	50.99	**41.55**
K0.0 V0.5	23.80	43.14	43.32	46.28	39.42	22.97	29.18	22.70	27.13	74.50	90.50	41.74	5.00	67.50	57.23	54.30	**43.04**
K0.0 V0.7	24.19	42.78	43.92	45.82	39.11	22.53	26.92	22.52	26.12	74.00	90.36	39.88	5.50	65.50	59.18	56.05	**42.77**
K0.5 V0.5	23.40	46.63	42.98	46.28	39.27	23.13	28.29	22.78	27.07	74.00	90.58	39.97	5.00	67.00	55.54	53.46	**42.65**
K0.7 V0.7	24.10	40.85	40.88	44.93	38.03	22.36	24.02	21.90	24.78	70.50	90.04	37.77	5.25	63.00	54.12	52.86	**40.96**

Evaluation: Compression Efficiency

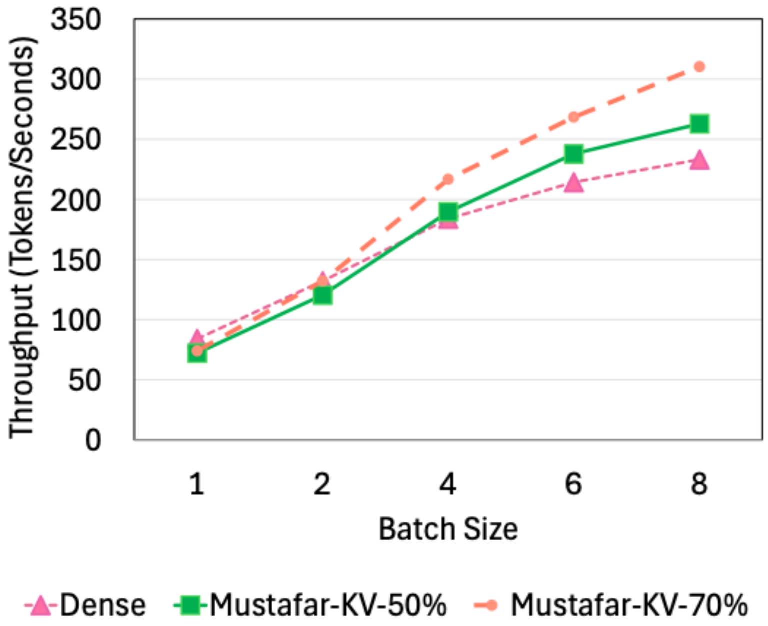
- Mustafar achieves higher accuracy with better compression compared to ThinK.



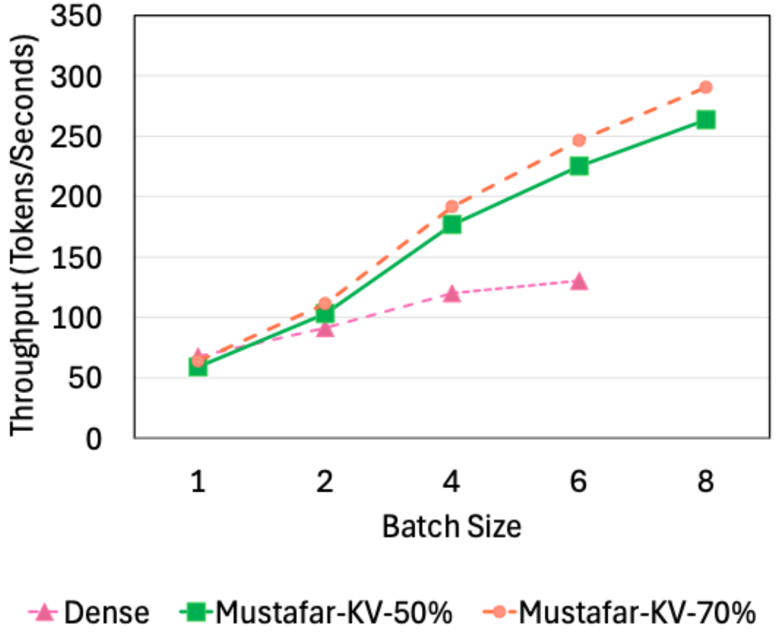
Compression ratio – accuracy comparison

Evaluation: Token Throughput

- Mustafar achieves higher throughput compared to dense with FlashAttention-2.
- KV cache compression allows larger batch size, increasing throughput even more.



Llama-2 7B Throughput



Llama-3 8B Throughput

Thank You

See you at the Session!

Thu 4 Dec 11 a.m. — 2 p.m., Exhibit Hall C,D,E

dhjoo98@umd.edu

Paper & Code

This work is supported by
National Science Foundation (NSF).

Backup Slides

Evaluation: TTFT and Decode Speed

Table 14: Decode speed comparison with dense inference

Model	KV Format	TTFT	Decode Speed (decode 512)	Decode Speed (decode 1024)	Decode Speed (decode 2048)
Llama2	Dense	1.396 sec	88.685 tokens / sec	88.512 tokens / sec	79.185 tokens / sec
	Mustafar K0.5 V0.5	2.532 sec	89.452 tokens / sec	89.514 tokens / sec	85.687 tokens / sec
	Mustafar K0.7 V0.7	2.249 sec	96.386 tokens / sec	97.436 tokens / sec	95.120 tokens / sec
Llama3	Dense	2.769 sec	61.993 tokens / sec	61.220 tokens / sec	59.242 tokens / sec
	Mustafar K0.5 V0.5	3.269 sec	78.434 tokens / sec	83.768 tokens / sec	83.303 tokens / sec
	Mustafar K0.7 V0.7	3.151 sec	84.065 tokens / sec	88.293 tokens / sec	89.699 tokens / sec

- TTFT is delayed due to prefill KV cache pruning and compression.
- Quickly amortized by accelerated decode.