Token Sparse Attention Efficient Long-Context Inference with Interleaved Token Selection

Dongwon Jo, Beomseok Kang, Jiwon Song, Jae-Joon Kim
Seoul National University

Arxiv 2602

present：唐承捷、傅申

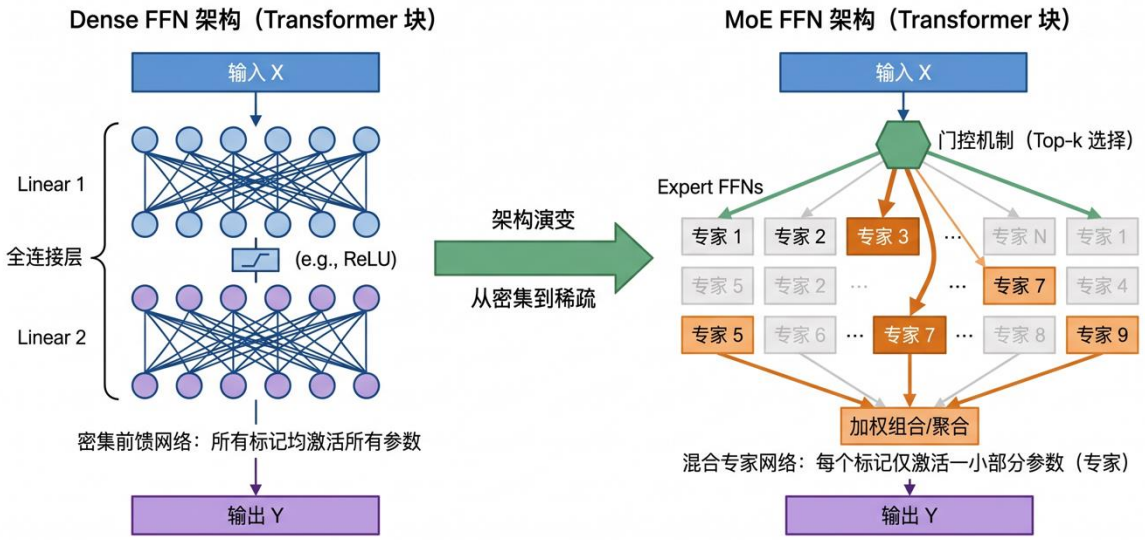
Context

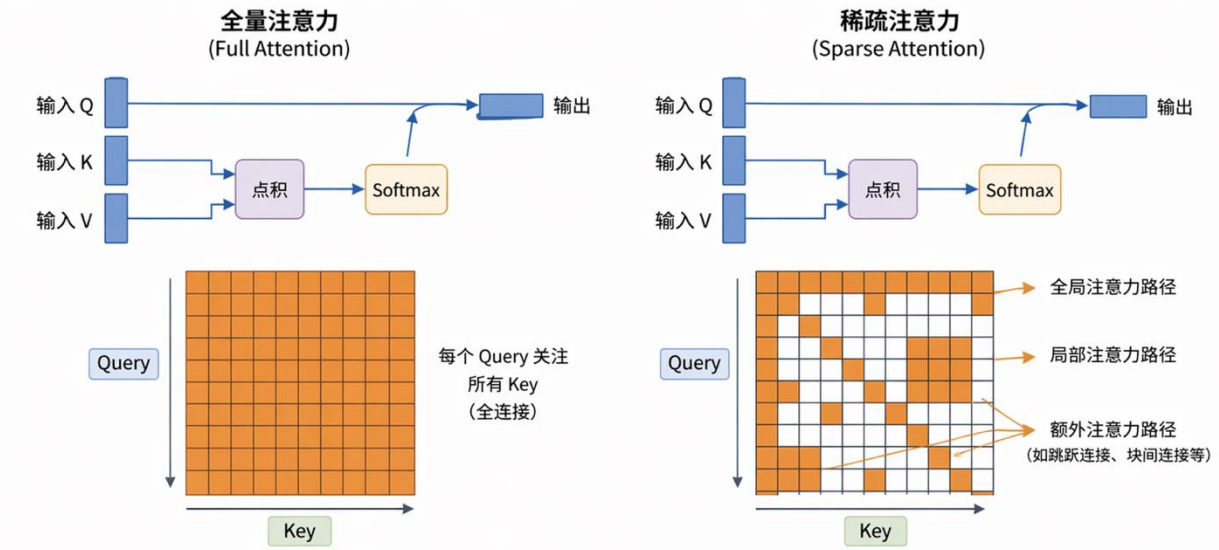
- ❑Why sparse attention?
- ❑Evolve of sparse attention algorithm
- ❑Motivation
- ❑Method
- ❑Evaluation
- ❑What’s next?

Why sparse attention?

- 模型算力**需求** >> 算力**硬件提升**
- 32k → 64k → 128k → 200k → 1M
- Agent

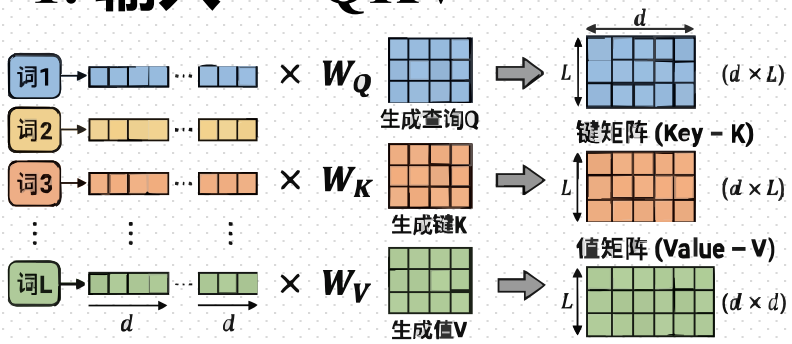
The solution for FFN is MoE，，，What about Attention?





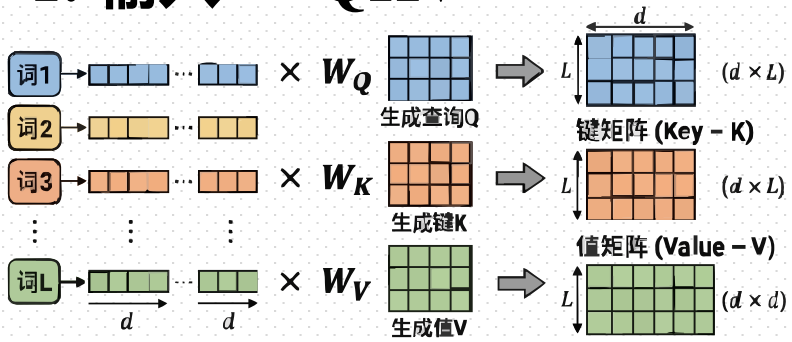
Why sparse attention? Long Context

1. 输入 → QKV

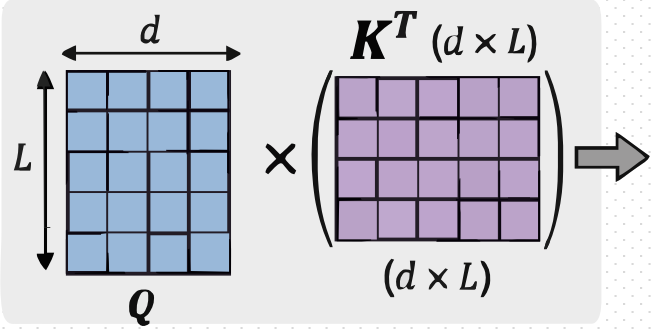


Why sparse attention? Long Context

1. 输入 → QKV

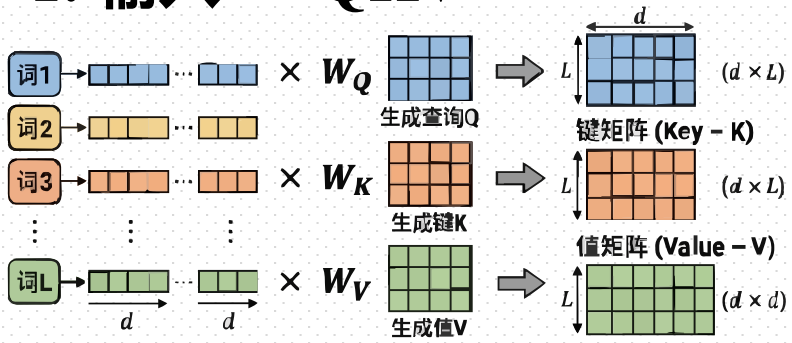


2. $Softmax(QK^T)V$ $O(L^2)$

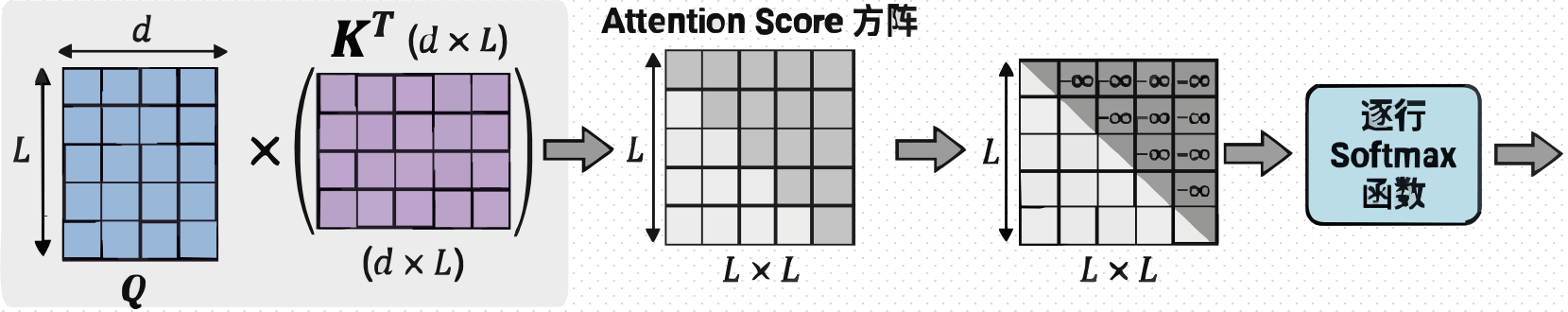


Why sparse attention? Long Context

1. 输入 → QKV

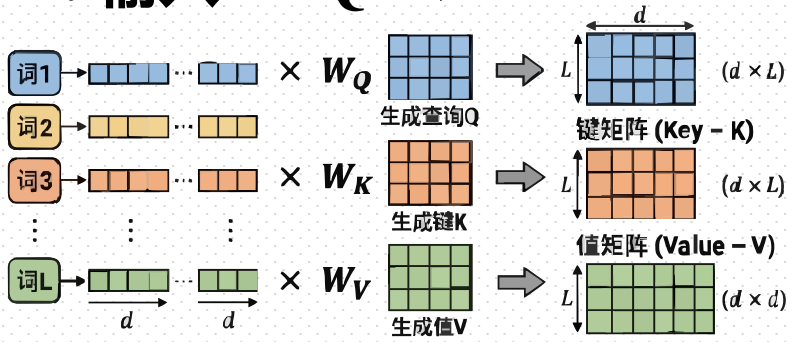


2. $Softmax(QK^T)V$ $O(L^2)$

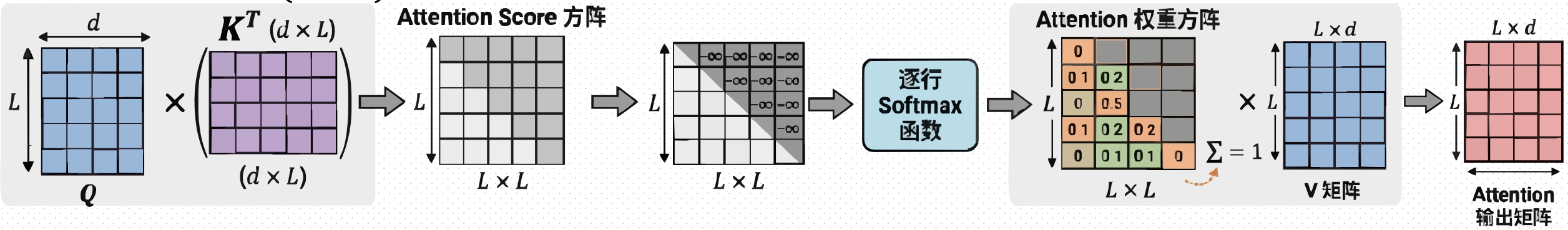


Why sparse attention? Long Context

1. 输入 → QKV

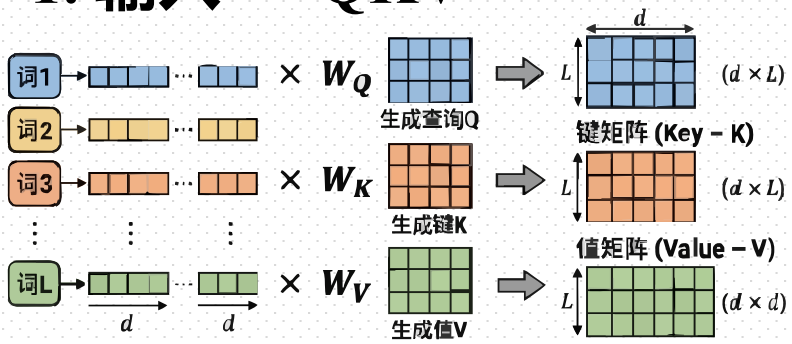


2. $Softmax(QK^T)V$ $O(L^2)$

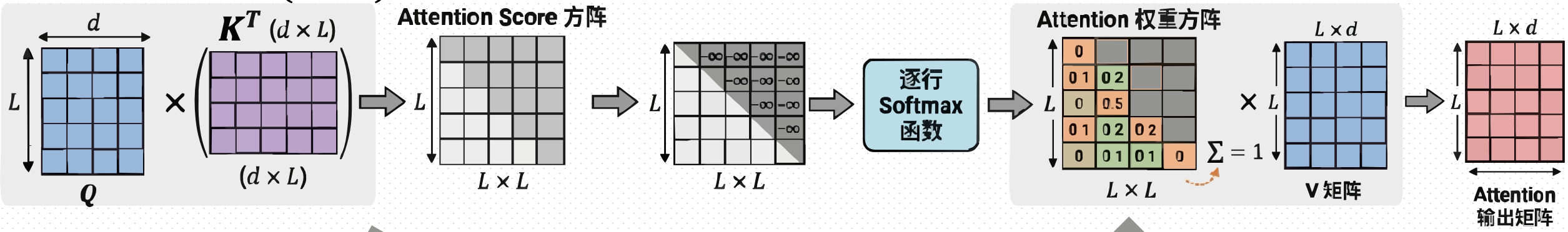


Why sparse attention? Long Context

1. 输入 → QKV



2. $Softmax(QK^T)V$ $O(L^2)$



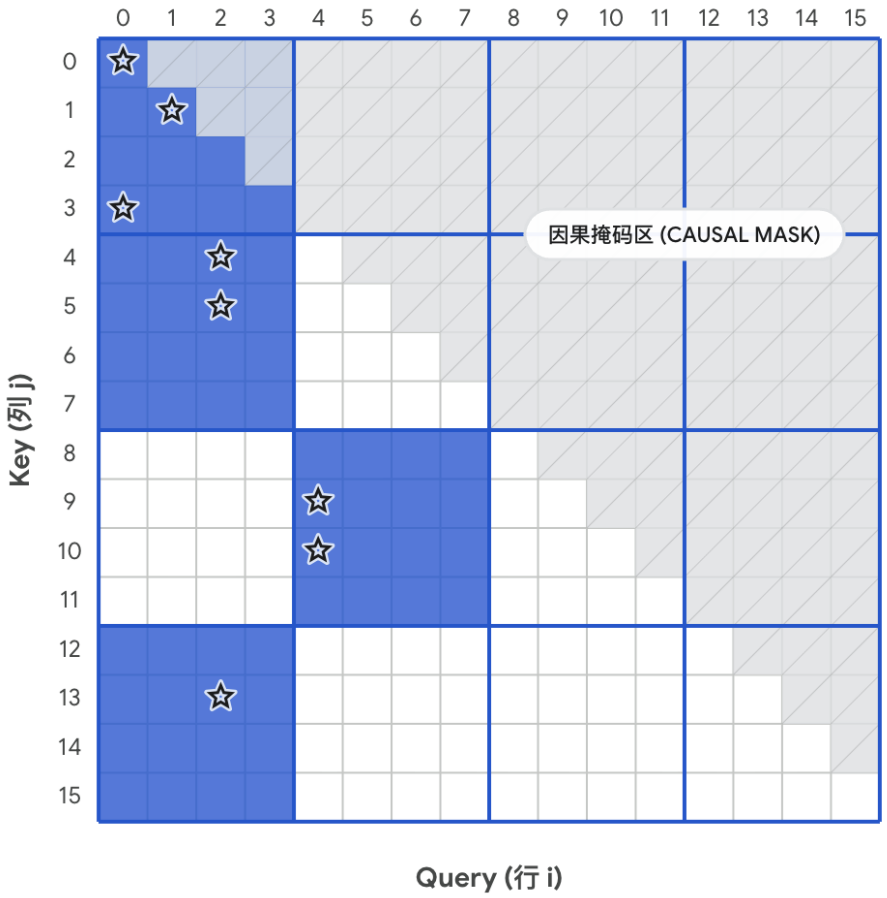
怎么选择token？和硬件有什么冲突？

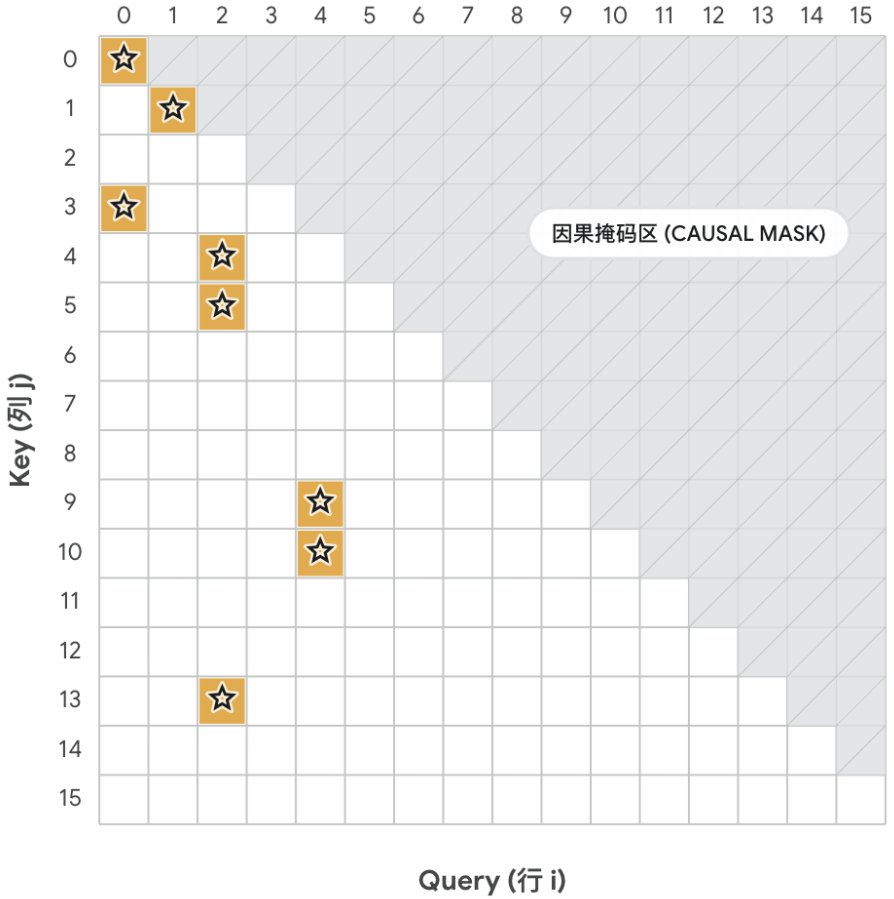
算法 **系统**

Evolution of Sparse Attention Algorithm

1. 粒度变化：Block → Token

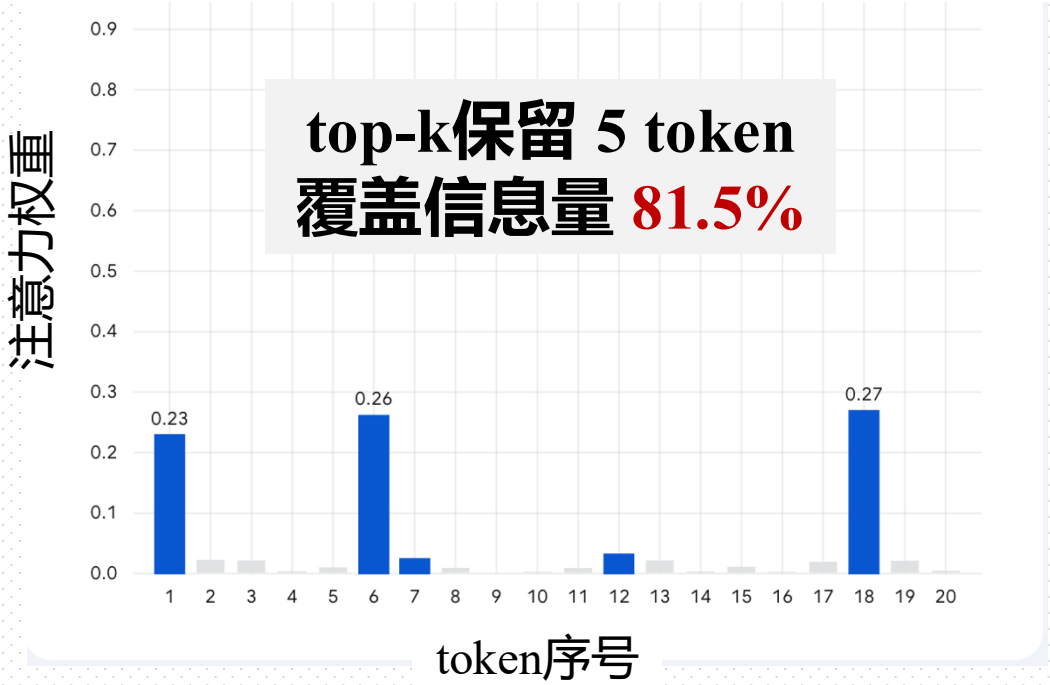
- Block：MInference1(NIPS 2407)、 FlexPrefill(ICLR 2410)、 xAttention(ICML 2503)、 NSA(ACL 2503)
- Token：DSA(2512)、 **Token Sparse Attention(2602)**



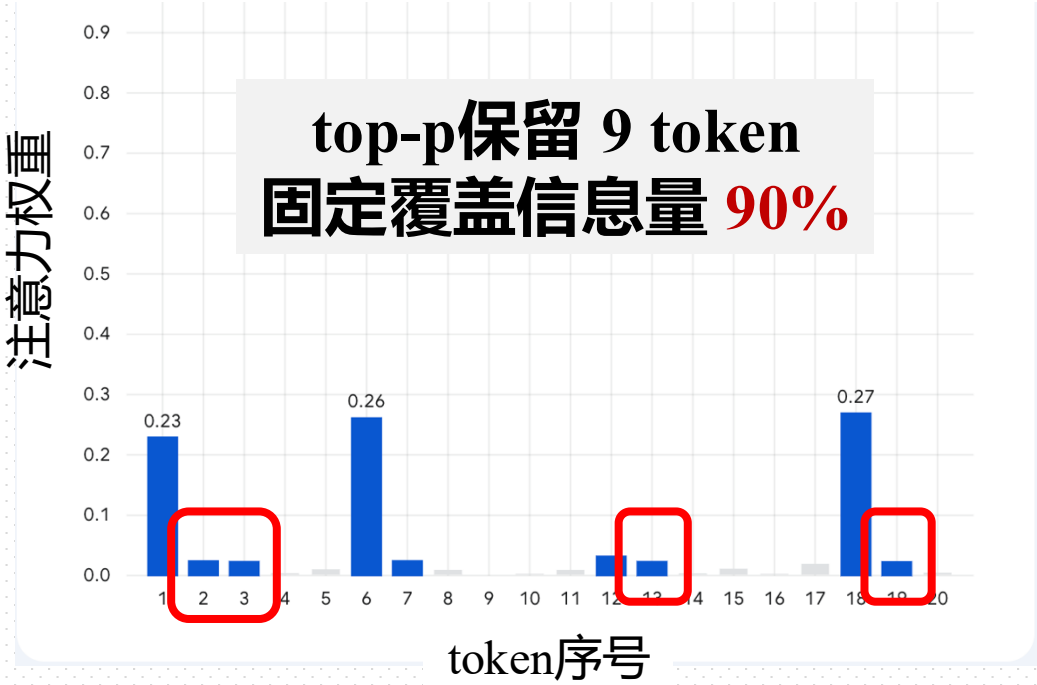


Evolution of Sparse Attention Algorithm

1. 粒度变化：Block → Token
2. 预算自适应：Top-k → Top-p



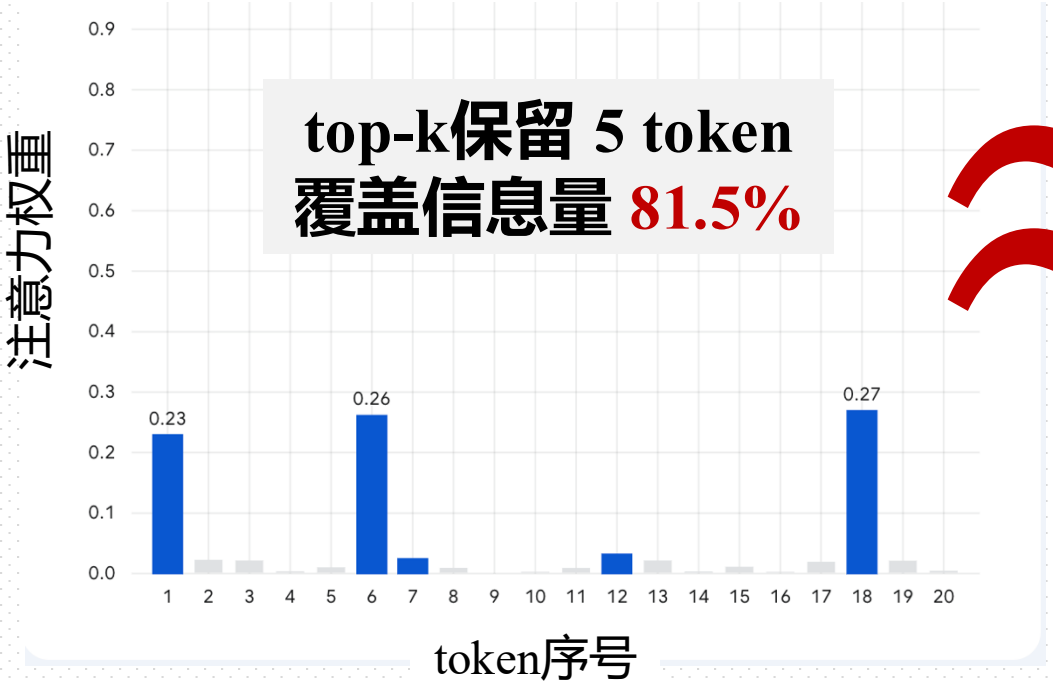
集中分布、top-k



集中分布、top-p

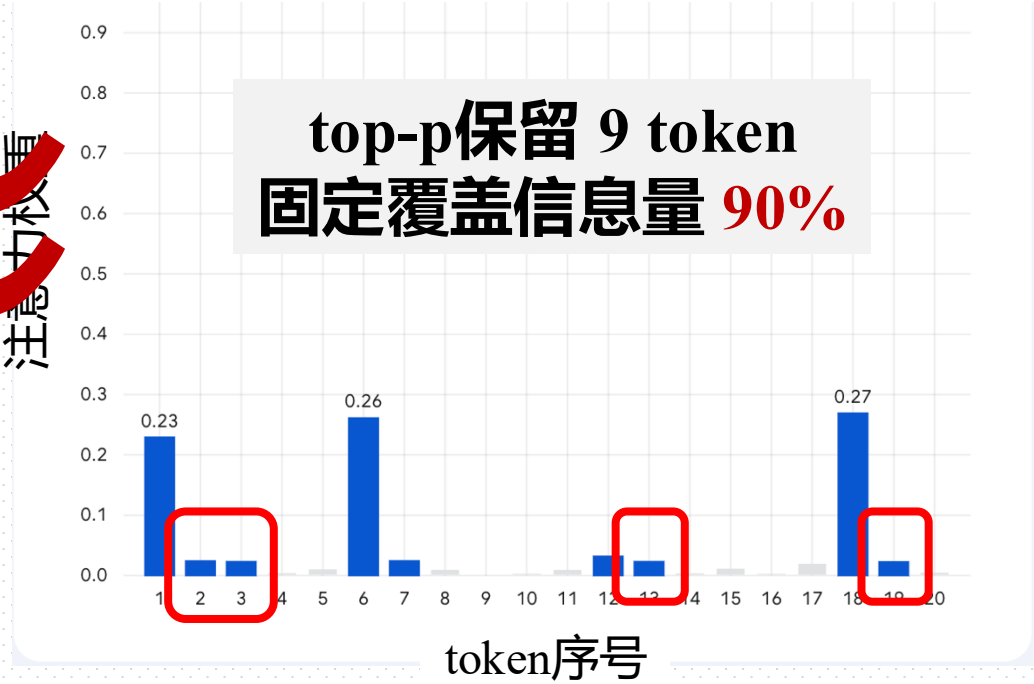
Evolution of Sparse Attention Algorithm

1. 粒度变化：Block → Token
2. 预算自适应：Top-k → Top-p



集中分布、top-k

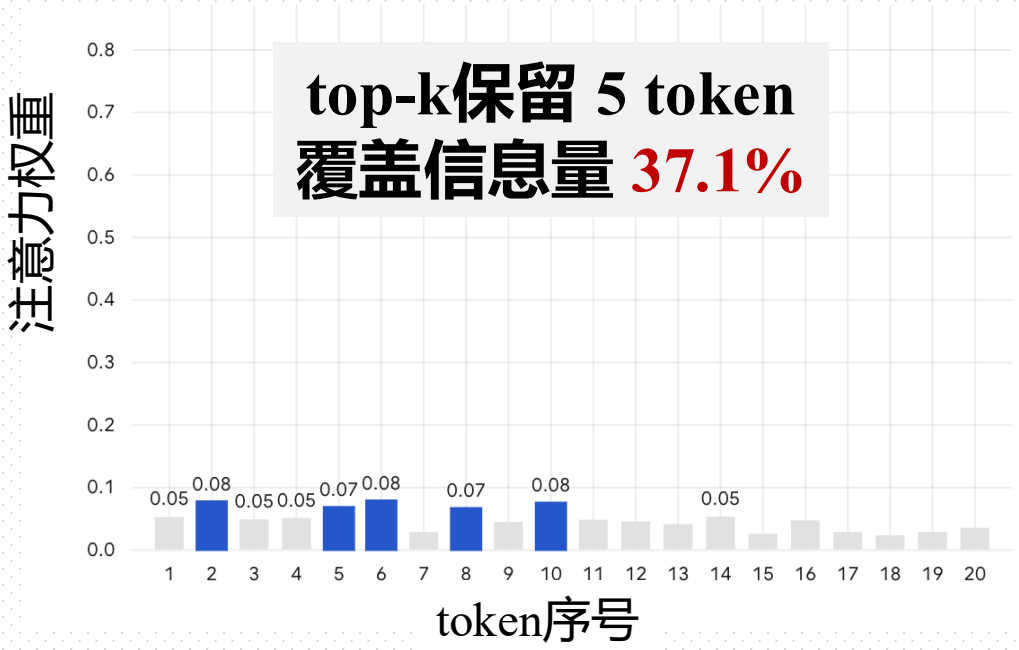
≈



集中分布、top-p

Evolution of Sparse Attention Algorithm

1. 粒度变化：Block → Token
2. 预算自适应：Top-k → Top-p

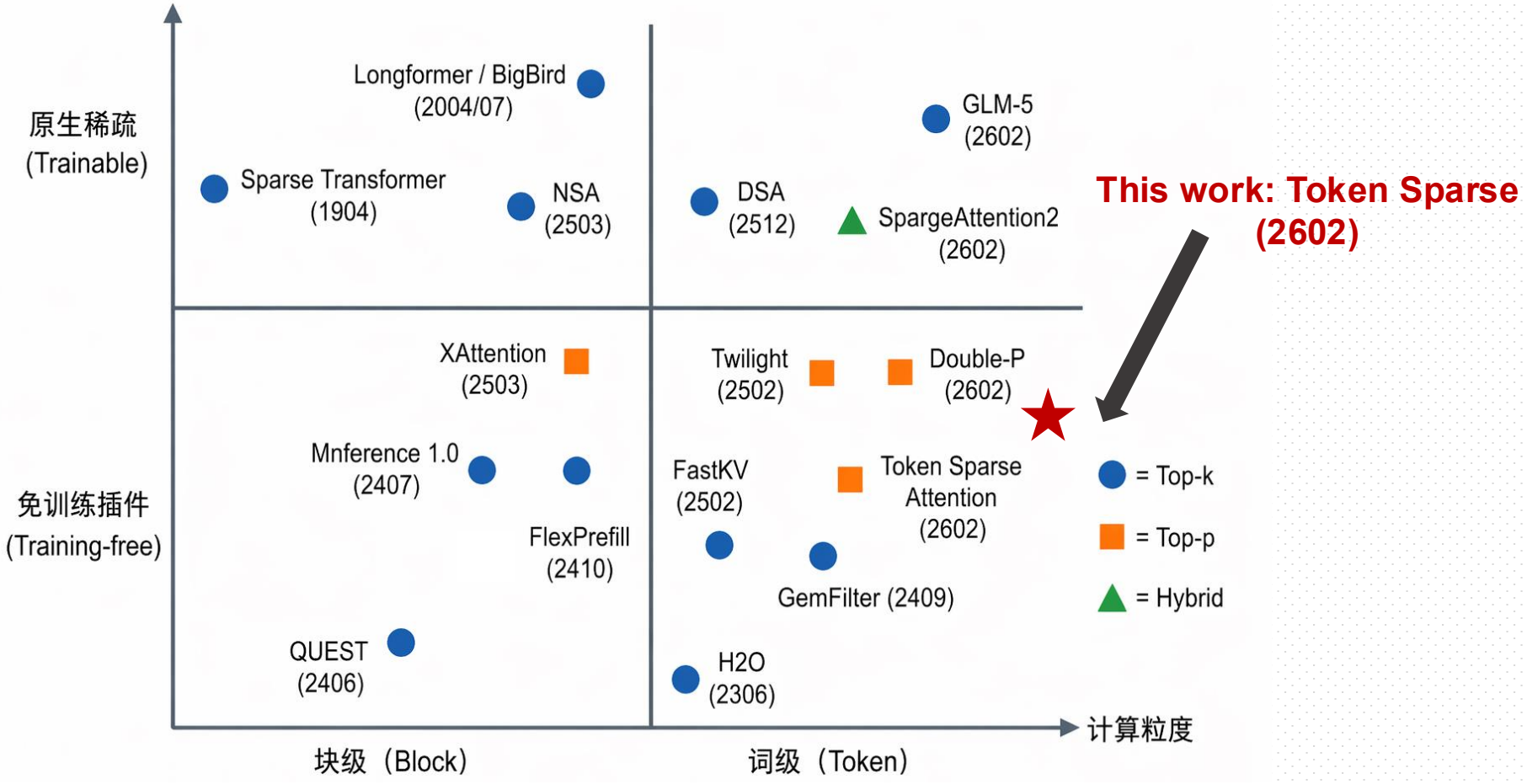


分散分布、top-k

分散分布、top-p

Evolution of Sparse Attention Algorithm

1. 粒度变化：Block → Token
2. 预算自适应：Top-k → Top-p



Motivation

1. 当前Token wise算法问题：Token Eviction

- Jo, D., Song, J., Kim, Y., and Kim, J.-J. Fastkv: Kv cache compression for fast long-context processing with tokenselective propagation. **同组前作**，(Arxiv 2502) (**ACL Findings 2026**)
- Shi, Z., Ming, Y., Nguyen, X.-P., Liang, Y., and Joty, S. Discovering the gems in early layers: Accelerating longcontext llms with 1000x input token reduction. (Arxiv 2409) (ICLR 25 reject) (**54 cite**)

Token Eviction

模型浅层筛出Token子集，在后续层中仅计算子集

Motivation

1. 当前Token wise算法问题： Token Eviction

- Jo, D., Song, J., Kim, Y., and Kim, J.-J. Fastkv: Kv cache compression for fast long-context processing with tokenselective propagation. **同组前作**，(Arxiv 2502) (**ACL Findings 2026**)
- Shi, Z., Ming, Y., Nguyen, X.-P., Liang, Y., and Joty, S. Discovering the gems in early layers: Accelerating longcontext llms with 1000x input token reduction. (Arxiv 2409) (ICLR 25 reject) (**54 cite**)

Token Eviction

模型浅层筛出Token子集，在后续层中仅计算子集

算法　　系统

2. 探索/改进方向：怎么选择Token？和硬件有什么冲突？

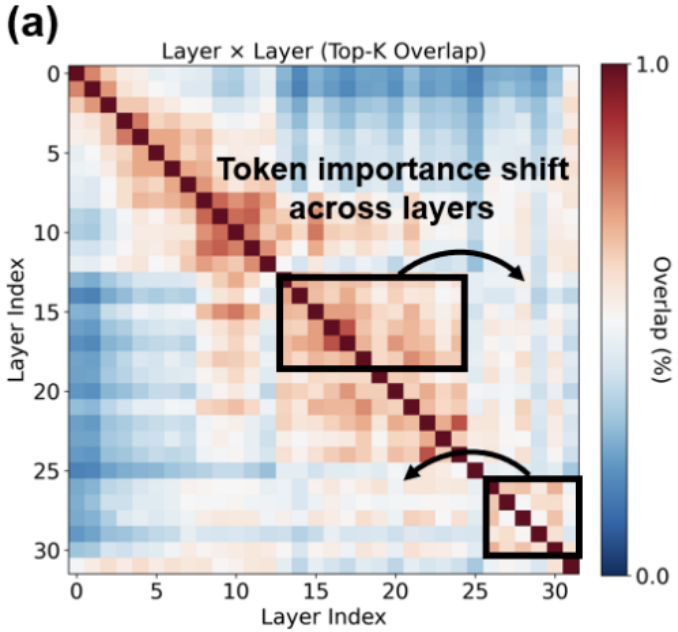
Motivation：Layer差异

1. 当前Token wise算法问题： Token Eviction

模型浅层筛出Token子集，在后续层中仅计算子集

2. 本文探索/改进方向：怎么选择Token?

Layer 差异（左a）：浅层筛出的Token，重要性变化?



- Experiment on LLaMA-3.1-8B-Instruct

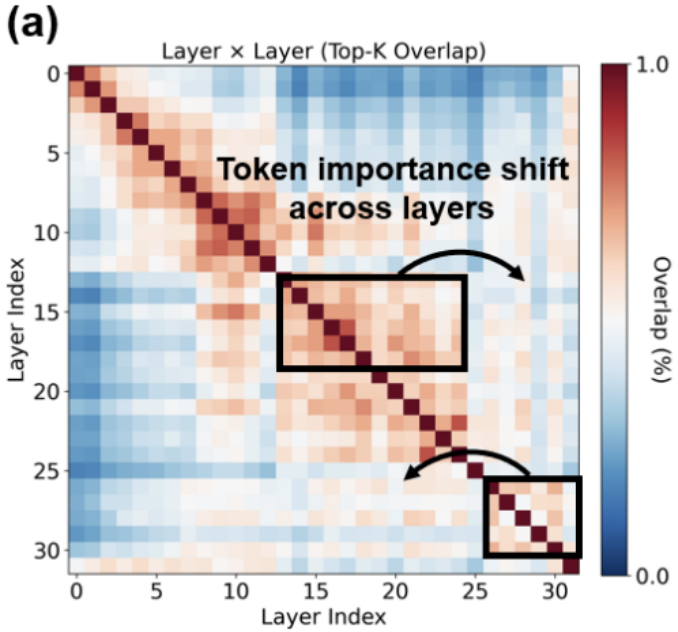
Motivation：Layer差异

1. 当前Token wise算法问题： Token Eviction

模型浅层筛出Token子集，在后续层中仅计算子集

2. 本文探索/改进方向：怎么选择Token?

Layer 差异（左a）：浅层筛出的Token，重要性变化？ YES



- Experiment on LLaMA-3.1-8B-Instruct

Motivation：Head差异

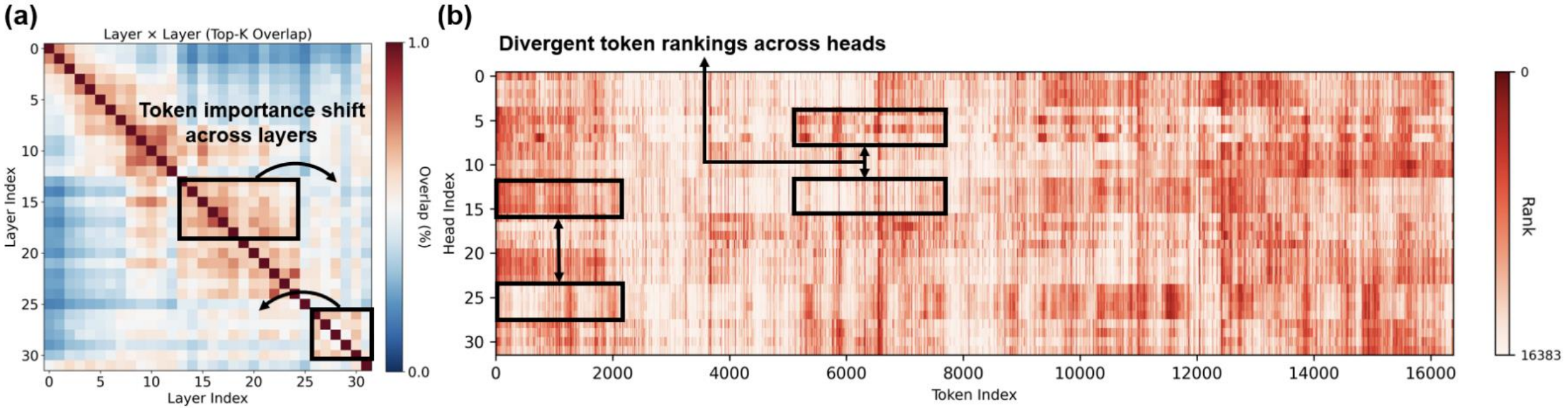
1. 当前Token wise算法问题： Token Eviction

模型浅层筛出Token子集，在后续层中仅计算子集

2. 本文探索/改进方向：怎么选择Token?

Layer 差异（左a）：浅层筛出的Token，重要性变化？ YES

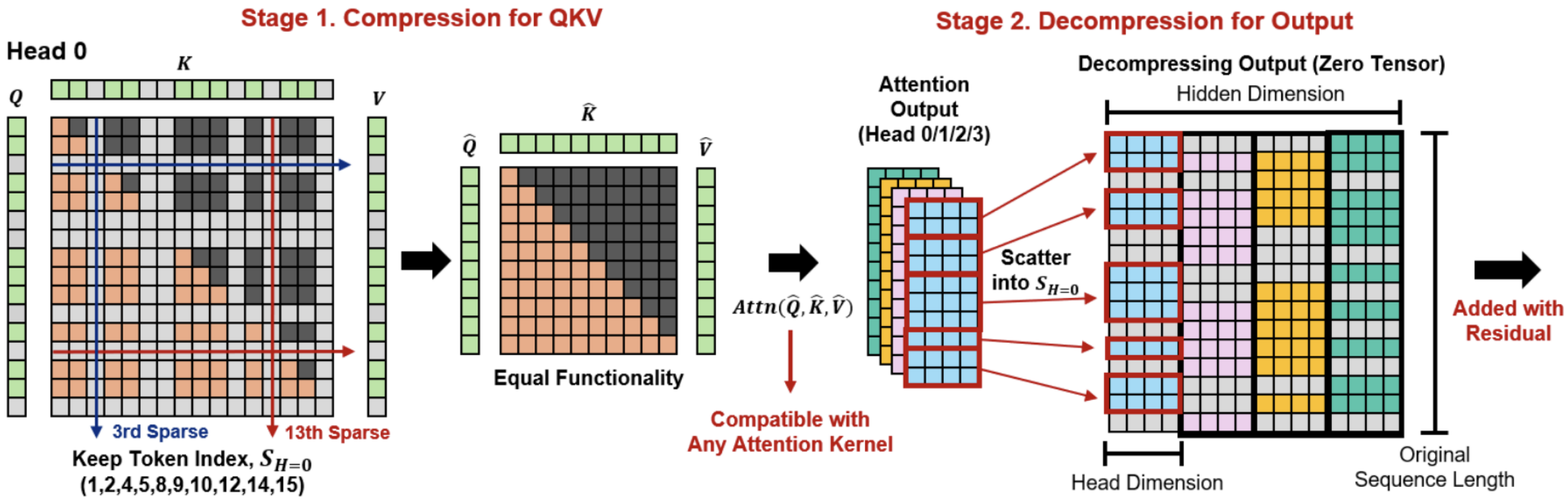
Head 差异（右b）：不同的Head会关注不同的Token？ YES



- Experiment on LLaMA-3.1-8B-Instruct

Method：Head选择重要Token

Head 差异：



选择重要token | Gather到物理连续 | Attention计算(兼容) | 计算结果Scatter

Method：Head选择重要Token

Head 差异：如何选择重要token

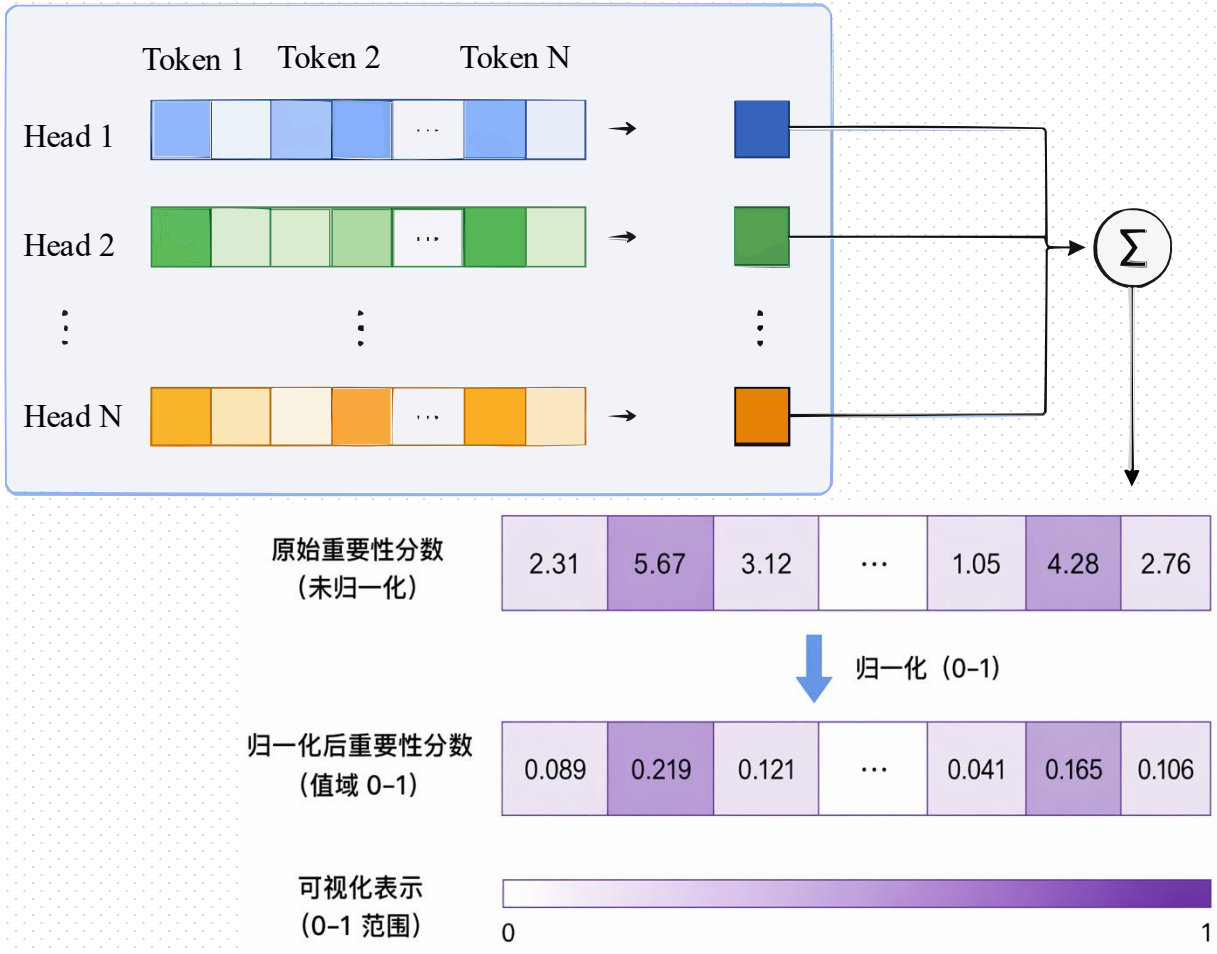
选择重要token | Gather到物理连续 | Attention计算(兼容) | 计算结果Scatter

保留多少token

聚合token在所有head贡献度+归一化

排序

丢弃最不重要的



Method：Head选择重要Token

Head 差异：如何选择重要token

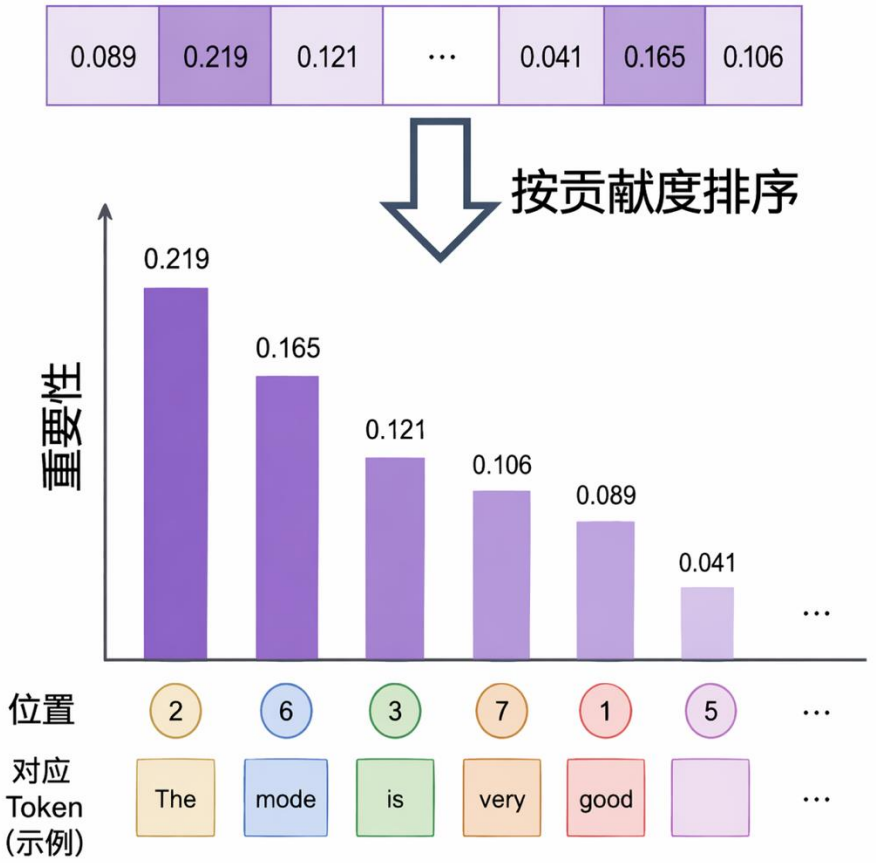
选择重要token | Gather到物理连续 | Attention计算(兼容) | 计算结果Scatter

保留多少token

聚合token在所有head贡献度+归一化

排序

丢弃最不重要的



Method：Head选择重要Token

Head 差异：如何选择重要token

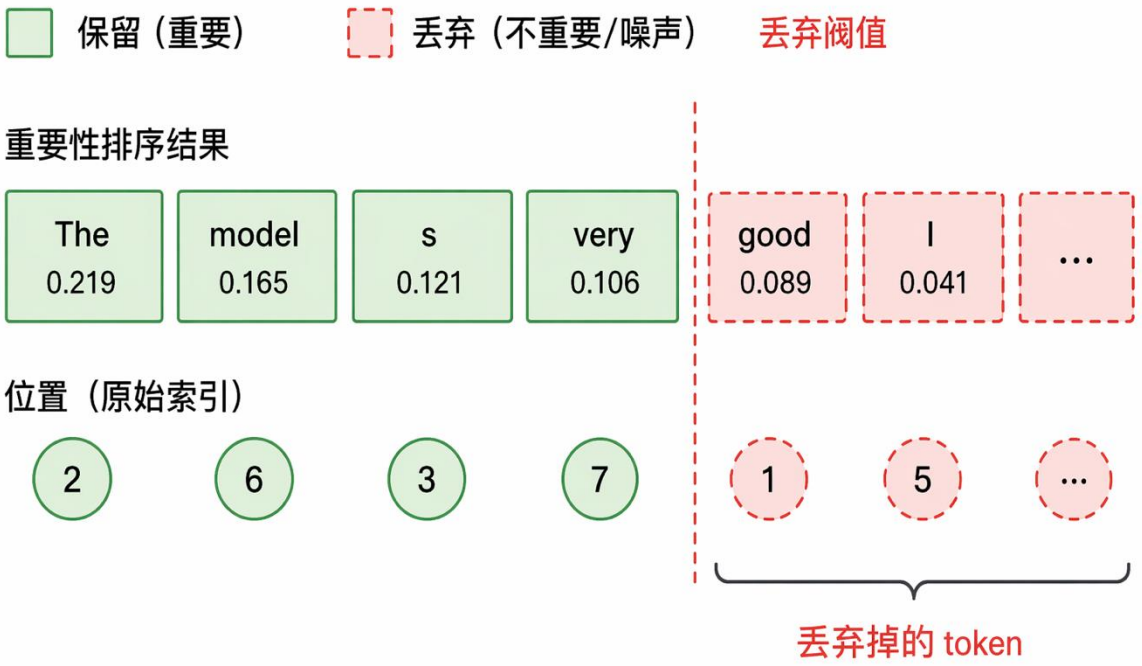
选择重要token | Gather到物理连续 | Attention计算(兼容) | 计算结果Scatter

保留多少token

聚合token在所有head贡献度+归一化

排序

丢弃最不重要的



Method：Head选择重要Token

Head 差异：如何选择重要token

选择重要token → Gather到物理连续 → Attention计算(兼容) → 计算结果Scatter

保留多少token

- 聚合token在所有head贡献度+归一化
- 排序
- 丢弃最不重要的

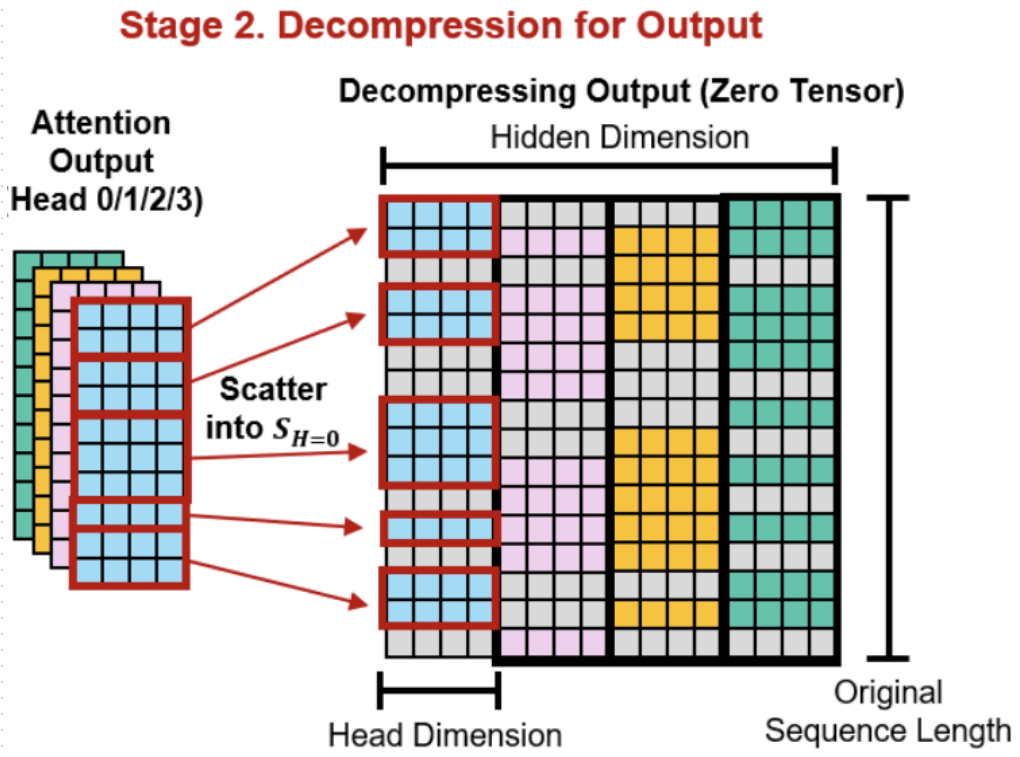
保留哪些token

- 每个head选择 top-k

全Head重要token

Method：Head计算结果写回

保留token | Gather到物理连续 | Attention计算(兼容) | **计算结果Scatter**



- 创建一个 全0完整序列
- 计算结果放回原位
- 残差连接

Method：Layer差异

Layer 差异：选择哪些层做稀疏

引入Inter-Layer Representation Drift，计算层间差异

$$R_\ell = \mathrm{E}_t\left[\frac{||h_{\ell+1,t} - h_{\ell,t}||_2}{||h_{\ell,t}||_2 + \epsilon}\right]$$

token 跨层差异

归一化

Method：Layer差异

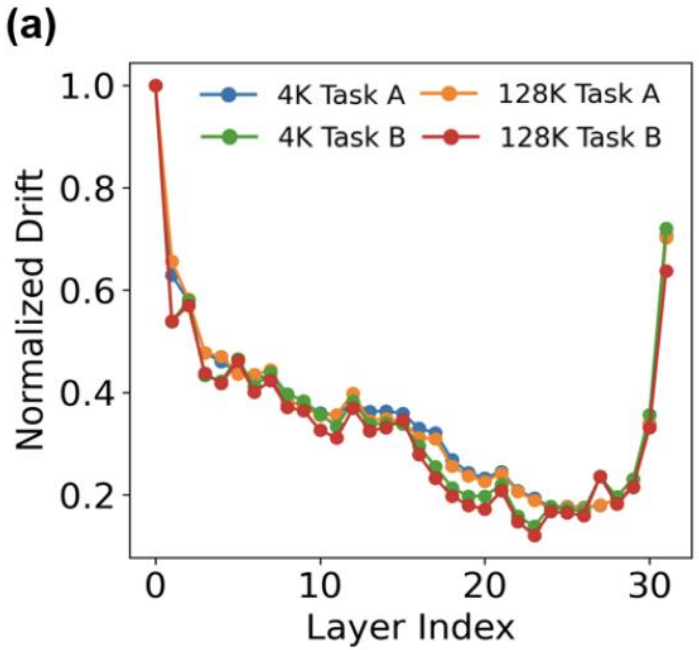
Layer 差异：选择哪些层做稀疏

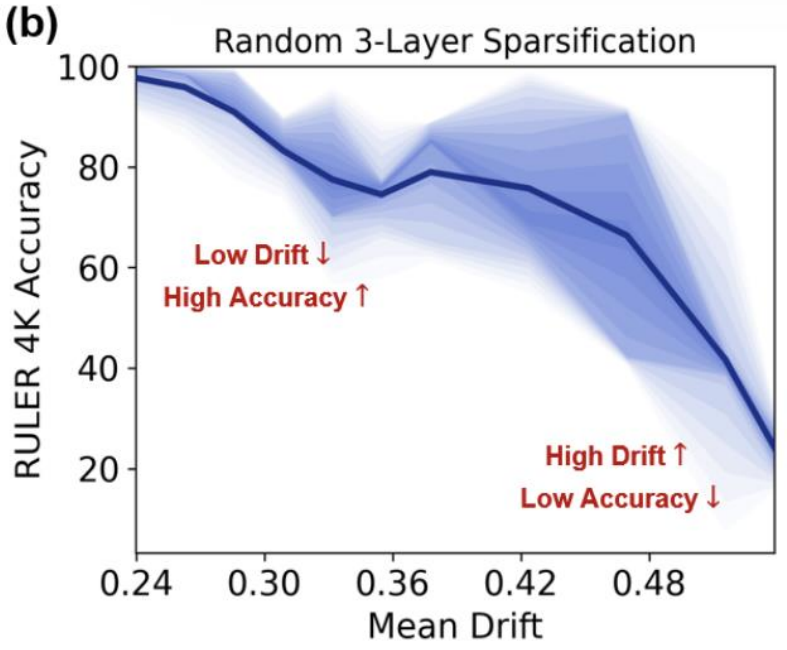
引入Inter-Layer Representation Drift，计算层间差异

$$R_\ell = \mathrm{E}_t\left[\frac{||h_{\ell+1,t} - h_{\ell,t}||_2}{||h_{\ell,t}||_2 + \epsilon}\right]$$

token 跨层差异

归一化





不同任务不发生变化

高drift确实影响精度

- Experiment on LLaMA-3.1-8B-Instruct

Evaluation: Setup

- **Models:**
  - LLaMA 3.1 8B Instruct
  - Mistral Nemo 12B Instruct
- **Benchmarks:**
  - RULER
  - InfiniteBench
  - LongBench
- **Hardware: A100 80G**
- **Baselines:**
  - **Dense:** FlashAttention
  - **Sparse:**
    - **Structured pattern:** Minference
    - **Block-sparse:** FlexPrefill
    - **Eviction-based token-sparse:** FastKV, GemFilter
- **Implementation:**
  - **Token scoring:** Triton; **Compression:** Torch
  - **Prefill:** Sparse Attn; **Decode:** Dense Attn
  - Underlying attn kernel remains unchanged

Eval: What We Aim to Validate?

- **Accuracy:**
  - Does TSA preserve model accuracy by avoiding the permanent eviction of tokens?
- **Efficiency:**
  - Does TSA effectively reduce the quadratic computational cost of the prefill phase for long contexts?
- **Compatibility:**
  - Is TSA designed as a highly flexible, “plug-and-play” module?

Eval: Accuracy – RULER

❑ **TSA provides complementary efficiency gains with negligible impact on accuracy.**

TSA Preserve the accuracy of underlying kernels

Context Length

Method	4K	8K	16K	32K	64K	128K	Avg.	Speedup
LLaMA-3.1-8B-Instruct ↓								
Flash Attention	95.82	92.77	91.02	84.87	83.43	74.15	87.01	×1.00
w/ Token Sparse	96.06	92.90	91.82	84.81	82.83	73.68	87.02	×1.36
Minference	93.46	92.29	91.01	85.34	83.19	73.63	86.49	×1.12
w/ Token Sparse	93.05	92.00	91.03	85.10	82.92	72.18	86.05	×1.38
FlexPrefill	95.48	92.71	91.40	87.20	83.05	73.75	87.27	×2.44
w/ Token Sparse	95.33	92.47	91.49	87.68	83.07	73.58	87.27	×2.76
Mistral-Nemo-12B-Instruct								
Flash Attention	95.19	92.29	85.60	64.86	47.98	19.68	67.60	×1.00
w/ Token Sparse	95.07	92.10	85.97	63.67	47.97	19.44	67.37	×1.22
Minference	92.52	91.02	84.29	65.18	46.52	19.00	66.42	×1.13
w/ Token Sparse	93.01	91.36	84.31	64.70	46.67	18.68	66.46	×1.28
FlexPrefill	94.79	93.13	86.62	64.58	49.30	20.54	68.16	×1.22
w/ Token Sparse	94.84	93.31	86.14	64.89	48.70	19.58	67.91	×1.33

(Speedup at 128K context)

Reliably increases the speedup across all methods

Eval: Accuracy – InfiniteBench

- TSA provides complementary efficiency gains with negligible impact on accuracy.

Marginal Accuracy Drop

Method	En.MC	En.QA	En.Dia	Retr.KV	Retr.N	Retr.P	Math.F	Code.D	Avg.
LLaMA-3.1-8B-Instruct									
Flash-Attention	64.19	27.64	19.00	55.20	98.47	97.80	24.00	20.56	50.86
w/ Token Sparse	65.07	28.01	17.00	55.00	98.47	97.46	25.71	20.30	50.88
Minference	67.25	26.80	18.50	51.60	97.46	97.80	22.29	19.54	50.16
w/ Token Sparse	65.94	27.61	18.50	49.60	97.46	97.80	21.43	19.29	49.70
FlexPrefill	68.12	27.37	11.00	50.80	99.15	96.95	22.57	20.30	49.53
w/ Token Sparse	67.25	27.60	14.00	46.80	98.98	96.27	22.86	20.05	49.23
Mistral-Nemo-12B-Instruct									
Flash-Attention	33.19	16.87	5.50	0.00	36.61	62.71	1.43	27.16	22.93
w/ Token Sparse	33.19	16.81	5.50	0.00	35.76	62.88	0.86	27.41	22.80
Minference	37.99	16.76	5.50	0.00	34.58	31.19	6.00	26.14	19.77
w/ Token Sparse	38.43	16.30	5.50	0.00	33.73	30.00	5.14	26.14	19.41
FlexPrefill	35.81	17.45	7.00	0.00	41.19	65.76	4.57	26.65	24.80
w/ Token Sparse	35.37	16.71	6.50	0.00	39.32	65.93	3.14	25.63	24.08

Eval: Accuracy – LongBench

- TSA provides complementary efficiency gains with negligible impact on accuracy.

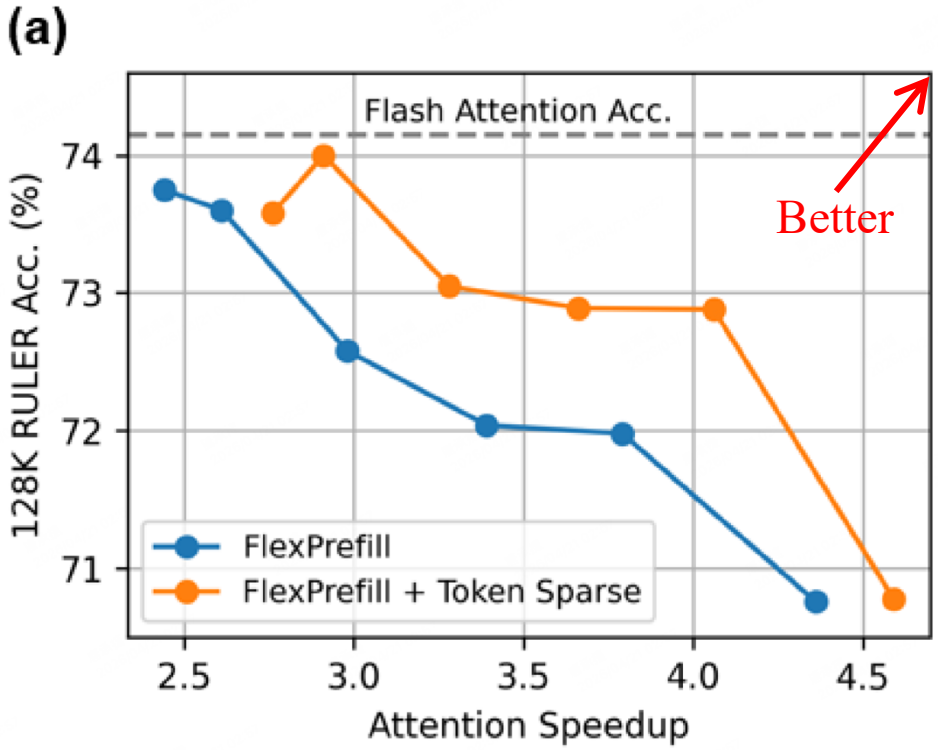
Marginal Accuracy Drop

Method	Single-Document QA			Multi-Document QA			Summarization			Few-shot Learning			Synthetic		Code		Avg.
	NrtvQA	Qasper	MF-en	HotpotQA	2WikiMQA	MuSiQue	GovReport	QMSum	MultiNews	TREC	TriviaQA	SAMSum	LCC	RB-P	PCount	PRe	
Flash-Attention	30.22	45.37	55.80	55.97	45.00	31.26	35.12	25.38	27.20	72.50	91.65	44.32	9.41	99.50	62.90	56.80	49.28
w/ Token Sparse	30.43	45.17	55.34	55.00	45.56	31.28	34.97	25.43	27.09	72.50	91.64	44.07	7.47	98.00	63.06	56.28	48.96
Minference	29.48	45.64	52.59	55.05	42.93	29.24	34.88	24.94	26.53	71.00	91.89	44.15	6.38	98.50	62.38	50.72	47.89
w/ Token Sparse	29.28	46.19	52.52	54.62	42.18	29.89	34.79	25.05	26.48	71.00	91.89	44.07	5.34	94.50	62.79	51.12	47.61
FlexPrefill	27.99	44.63	54.94	57.31	41.82	31.97	34.50	24.92	27.06	70.50	90.74	43.81	4.09	82.00	63.08	60.36	47.48
w/ Token Sparse	28.07	44.84	54.74	56.63	40.24	31.09	34.79	25.47	27.12	69.00	90.89	43.72	3.09	85.50	62.78	60.24	47.39

Eval: Efficiency – Accuracy-Speedup Tradeoffs

- **Applying TSA ($\tau = 0.005$) to FlexPrefill**
  - **Superior speedups at comparable accuracy levels**
  - **Such efficiency gain cannot be obtained by tuning FlexPrefill parameters**

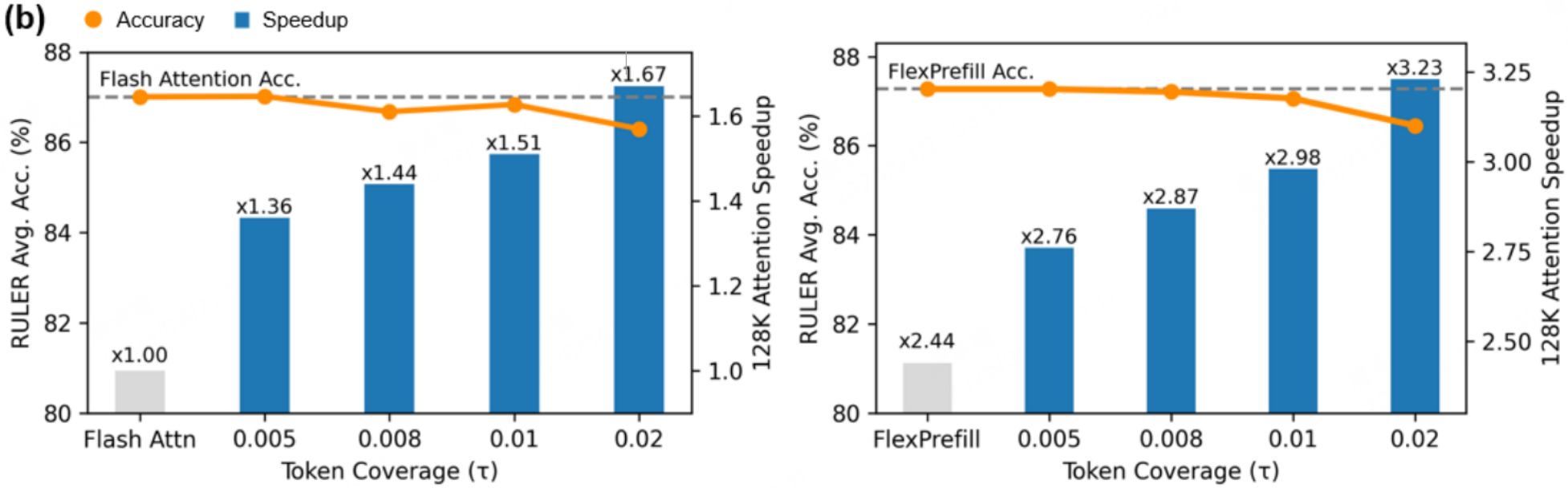
Pareto frontier obtained by varying FlexPrefill sparsity parameter



Eval: Efficiency – Accuracy-Speedup Tradeoffs

❑Effect of τ

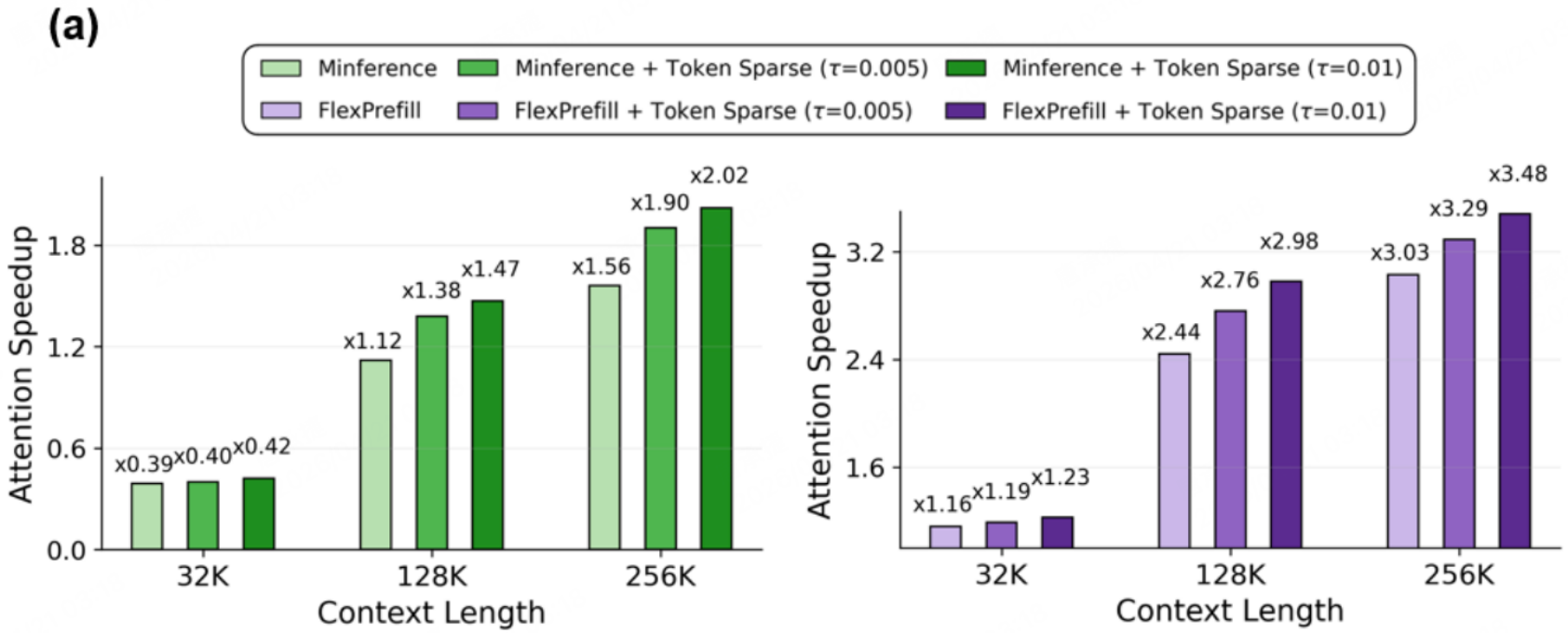
- ❖Higher τ → More aggressive sparsification → Higher Speedup
- ❖Compromise little model performance



Eval: Efficiency – Across Sequence Lengths

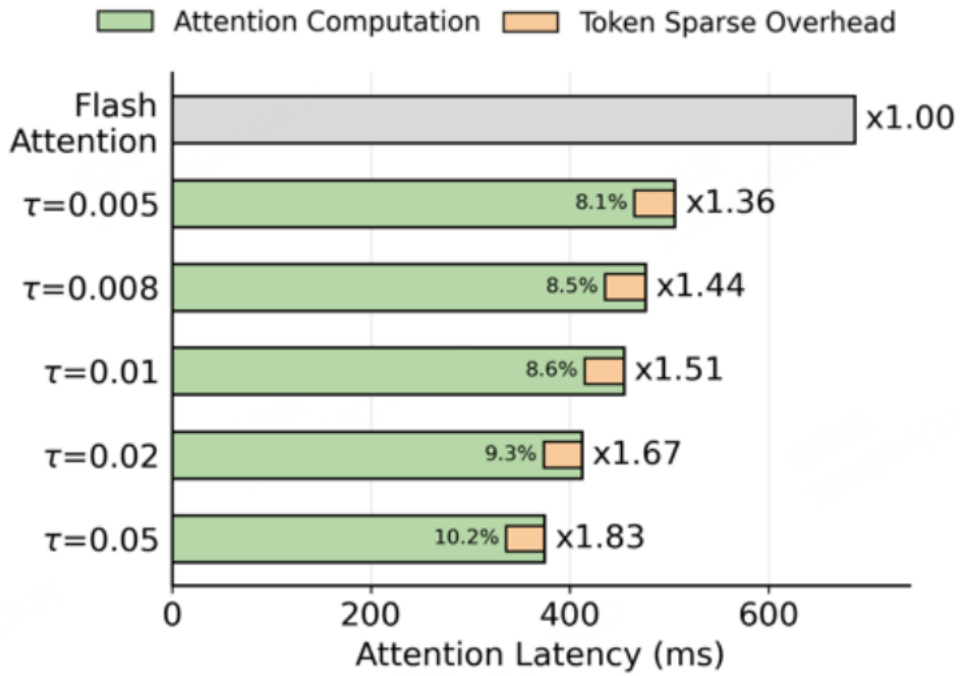
- **TSA becomes increasingly effective in longer-context regimes**
  - **Larger speedups at 128K / 256K**
  - **Sparsity increases steadily**

Sparsity	4K	8K	16K	32K	64K	128K
τ=0.005	17.00%	21.11%	26.61%	28.44%	34.07%	54.44%
τ=0.010	28.02%	32.98%	39.55%	41.25%	47.32%	67.36%



Eval: Efficiency – Latency Breakdown

- **Significant acceleration while incurring a small, well-bounded overhead (< 11%)**
  - **Token scoring, Token indexing, Compression and Decompression**



Eval: Efficiency – Dynamic vs. Fixed Sparsity

- RULER accuracy under matched attention speedup
- Dynamically allocating the sparsity budget is more effective
- ($\tau = 0.005$) outperforms ($s = 0.3$) while preserving a higher sparsity
  - Flexible, dynamic per-layer budget

	Dynamic Sparsity		Fixed Sparsity	
Metric	τ=0.005	τ=0.010	s=0.3	s=0.5
Sparsity	54.44%	67.36%	50.96%	74.95%
Accuracy	87.02%	86.84%	86.91%	85.43%
Speedup	×1.36	×1.51	×1.32	×1.57

Eval: Compare with Eviction-Based

- TSA achieves highest accuracy under comparable speedups
- Better recognition of layer-wise & head-wise variance
  - Layer-wise: dynamic budget + reversible interleaving
  - Head-wise: fine-grained, head-specific token selection

Method	4K	8K	16K	32K	64K	128K	Avg.	Speedup
FlashAttn	95.82	92.77	91.02	84.87	83.43	74.15	87.01	1.0x
GemFilter	92.50	90.87	88.78	85.01	80.42	73.15	85.12	1.53x
FastKV	94.25	91.30	89.78	84.57	81.54	72.39	85.64	1.50x
Ours	95.82	92.71	91.46	84.81	83.00	73.25	86.84	1.51x

Conclusion & Discussion

❑Evaluation Summary

- ❖**Accuracy:** <1% degradation across all four benchmarks
- ❖**Efficiency:** up to 3.23x attention speedup, overhead < 11%
- ❖**Compatibility:** plug-and-play with FlashAttention, MInference, FlexPrefill

❑Takeaway

- ❖Dynamic, reversible token-level sparsification is an effective and orthogonal strategy for long-context prefill acceleration.

❑Open Questions

- ❖Validated on 8B/12B only — does it scale to 70B+?
- ❖Fixed global τ — could adaptive thresholds do better?

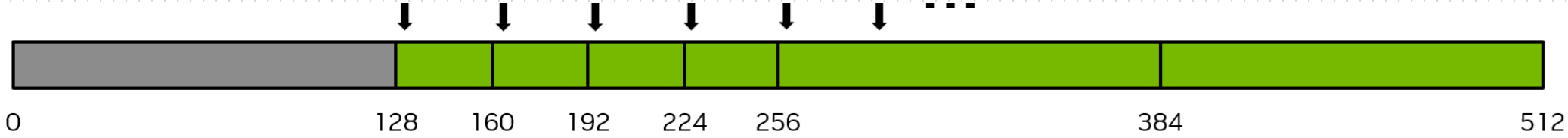
What's next? Memory System: micro

$$1\ \text{Token} = 1 \times 4096 \times 2\text{B} = 8\text{KB}$$

- **"现代" GPU: block read**

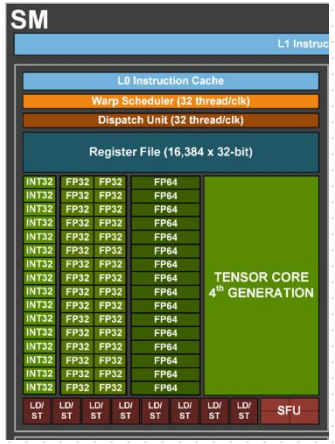
"Global memory is accessed via 32-byte memory transactions. "

---- CUDA Programming Guide 13.2

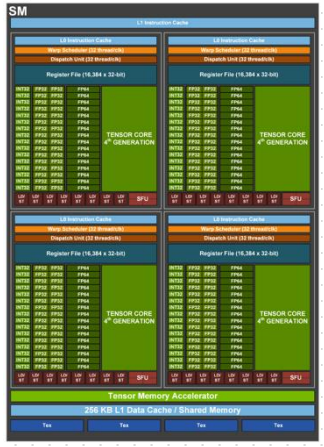


$1\ \text{Thread} = 32\ \text{B}$

$1\ \text{Warp} = 32 \times 32\text{B} = 1$



$1\ \text{SM} = 4 \times 1\text{KB} = 4\text{KB}$



- https://docs.nvidia.com/cuda/cuda-programming-guide/02-basics/writing-cuda-kernels.html#coalesced-global-memory-access
- Microbenchmarking NVIDIA's Blackwell Architecture: An in-depth Architectural Analysis (https://arxiv.org/pdf/2512.02189)

What's next? Memory System: micro

$$1\ \text{Token} = 1 \times 4096 \times 2\text{B} = 8\text{KB}$$

- **"现代" GPU: block read**

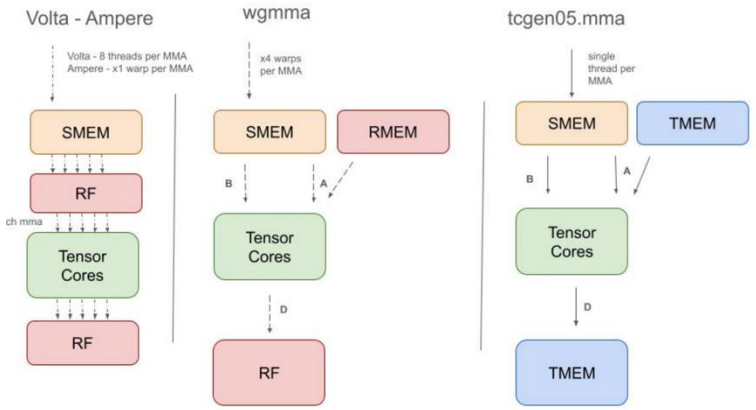
  "Global memory is accessed via 32-byte memory transactions. "

  ---- CUDA Programming Guide 13.2

  $1\ \text{Thread} = 32\ \text{B}$ $1\ \text{Warp} = 32 \times 32\text{B} = 1024\text{B}$ $1\ \text{SM} = 4 \times 1\text{KB} = 4\text{KB}$

- **Blackwell: TMEM 256KB on-chip memory per SM**

"TMEM achieves optimal efficiency at 64×64 element tiles (4KB for FP8 precision)"



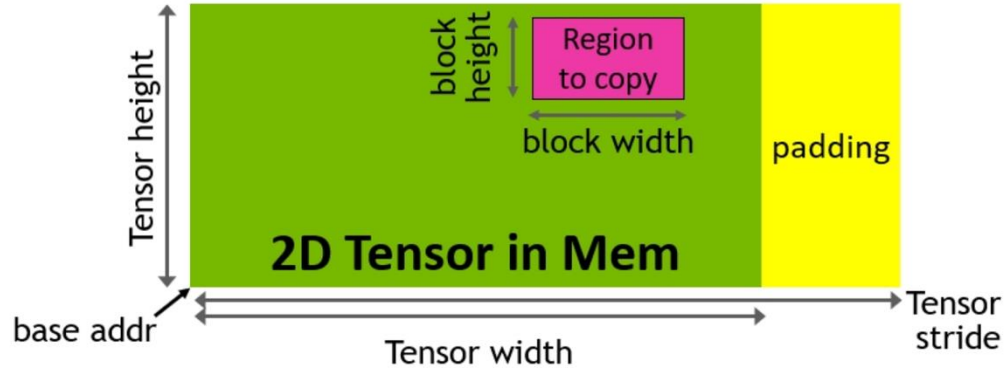
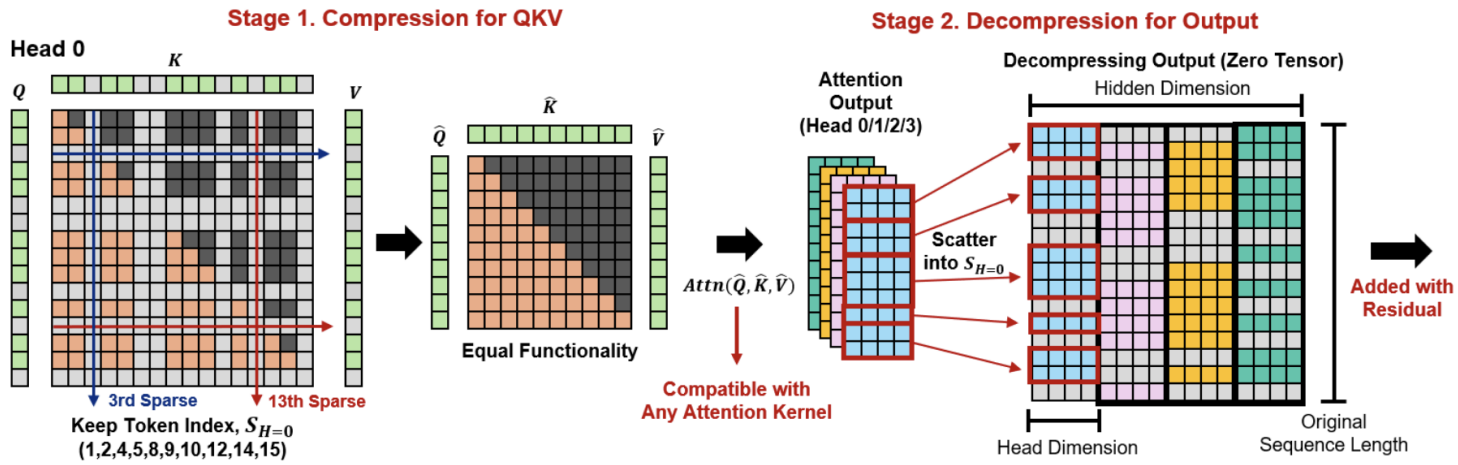


Figure 18. TMA Address Generation via Copy Descriptor

- https://docs.nvidia.com/cuda/cuda-programming-guide/02-basics/writing-cuda-kernels.html#coalesced-global-memory-access
- Microbenchmarking NVIDIA's Blackwell Architecture: An in-depth Architectural Analysis (https://arxiv.org/pdf/2512.02189)

What is overhead? Memory: Fused kernel

- Flash Attention
- Flash __?__ sparse Attention



选择重要token

连续访存

Gather到物理连续

稀疏访存

Attention计算(兼容)

连续访存

计算结果Scatter

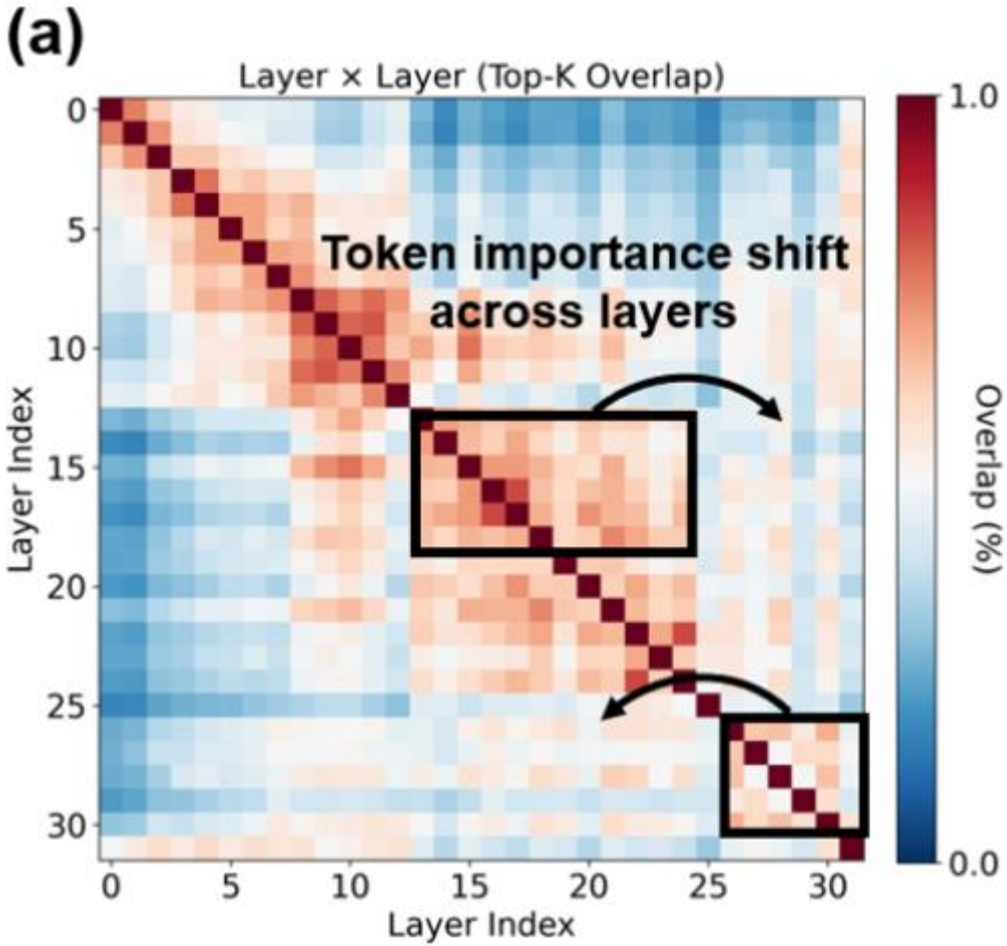
稀疏访存

fused Sparse Attn Kernel

稀疏访存

What's next? Layer similarity + Offload?

相邻层相似性 + token wise kv cache eviction机制



Select overhead << memory + comp save

Thanks