High Performance Computer Architecture
Institute of Computing Technology, CAS
Research Center for Intelligent Computing Systems

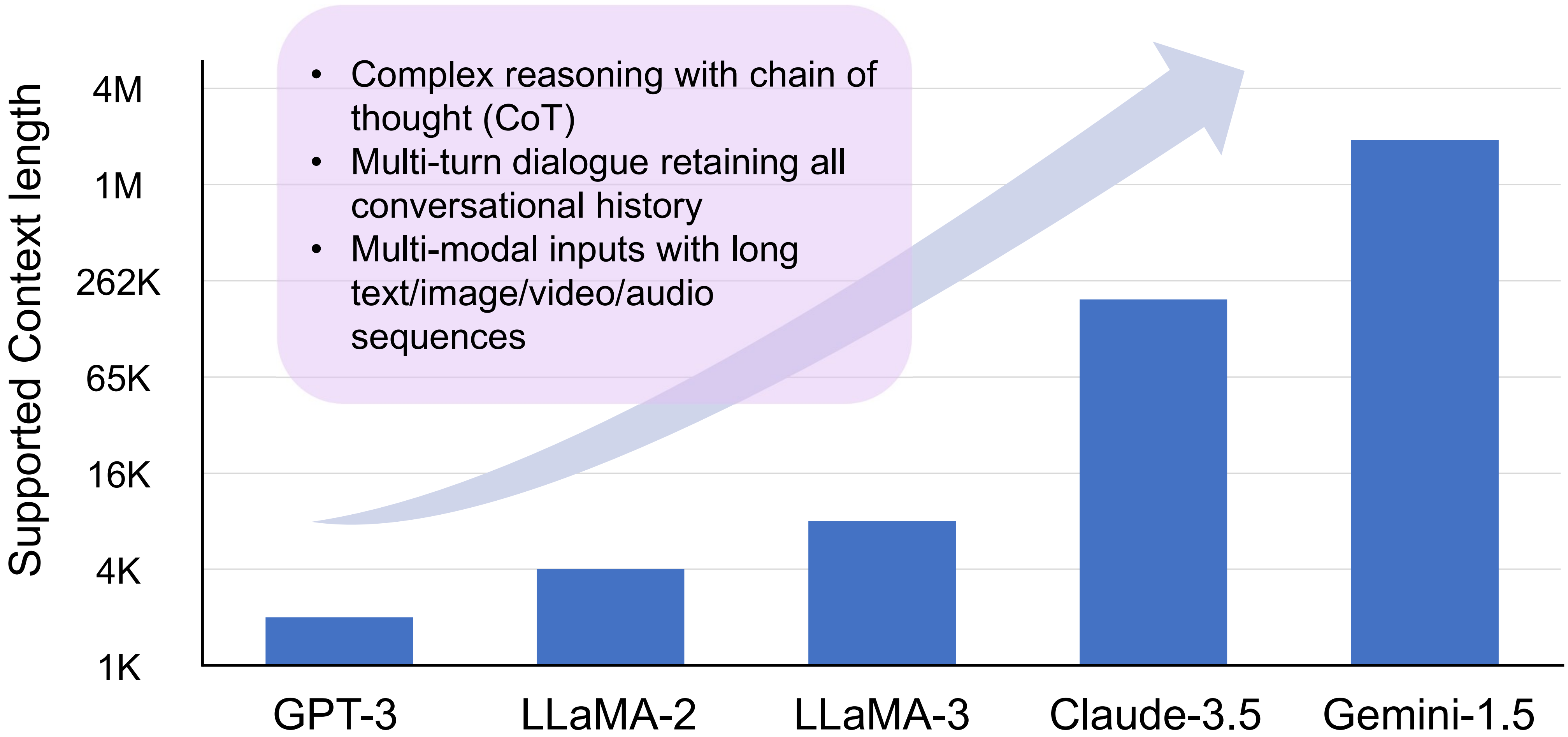
LAD: Efficient Accelerator for Generative Inference of LLM with Locality Aware Decoding

Haoran Wang, Yuming Li, Haobo Xu, Ying Wang, Liqi Liu, Jun Yang, Yinhe Han

Research Center for Intelligent Computing Systems
Institute of Computing Technology, Chinese Academy of Sciences

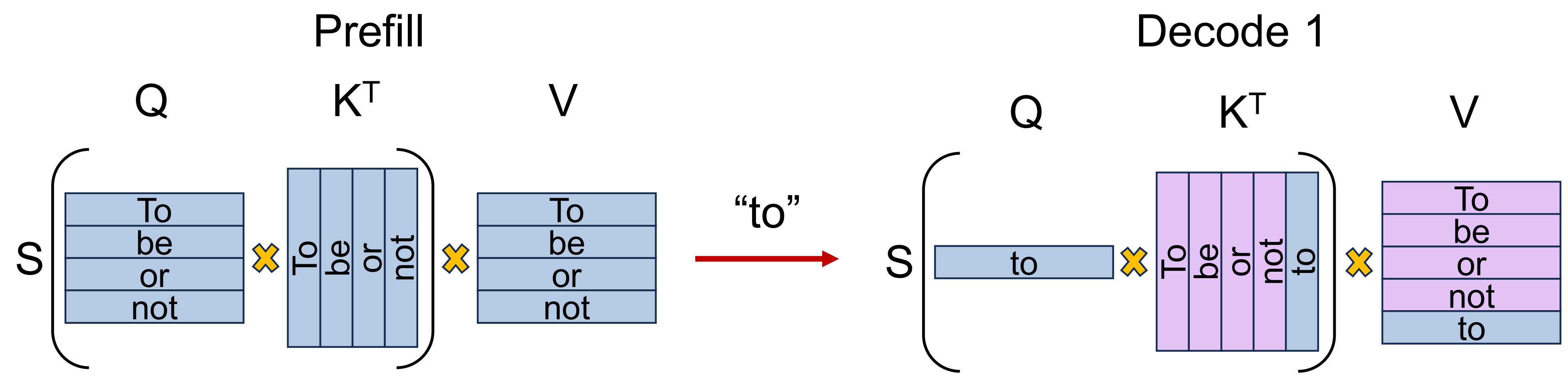
HPCA 2025

The pursuit of longer context length in LLM inference



The inefficiency of the KV cache

- Linearly growing size with respect to context length
- Repeated accesses during auto-regressive decoding



Restored from main memory

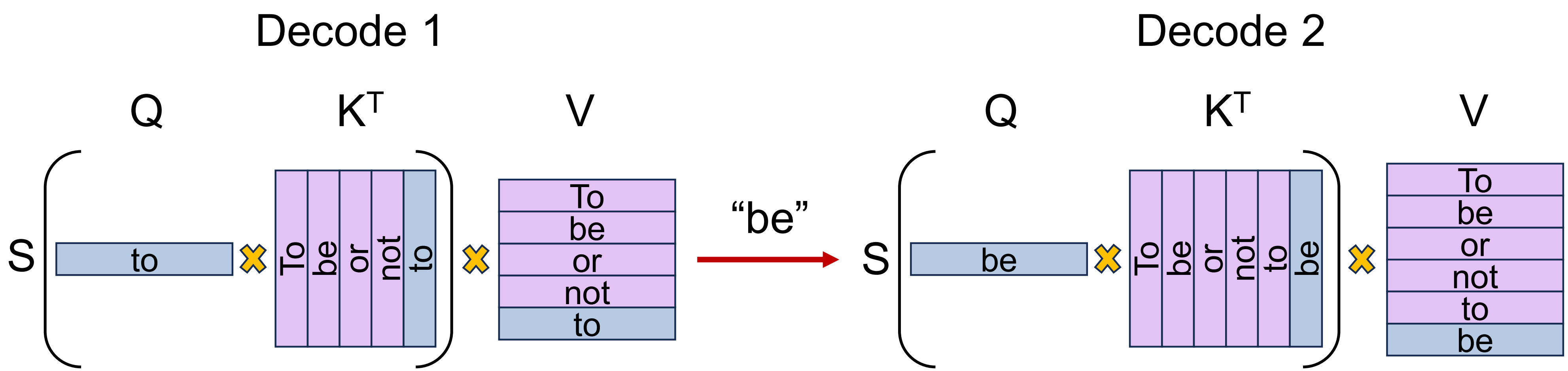
Computed in current step

Accumulated load times

	To	be	or	not
Decode 1	1	1	1	1

The inefficiency of the KV cache

- Linearly growing size with respect to context length
- Repeated accesses during auto-regressive decoding



Restored from main memory

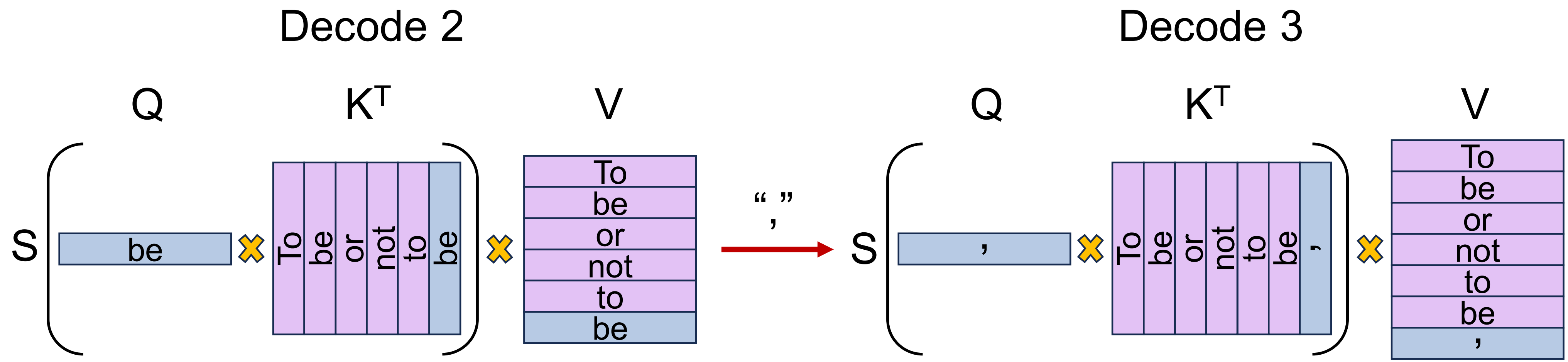
Computed in current step

Accumulated load times

	To	be	or	not	to
Decode 1	1	1	1	1	
Decode 2	2	2	2	2	1

The inefficiency of the KV cache

- Linearly growing size with respect to context length
- Repeated accesses during auto-regressive decoding



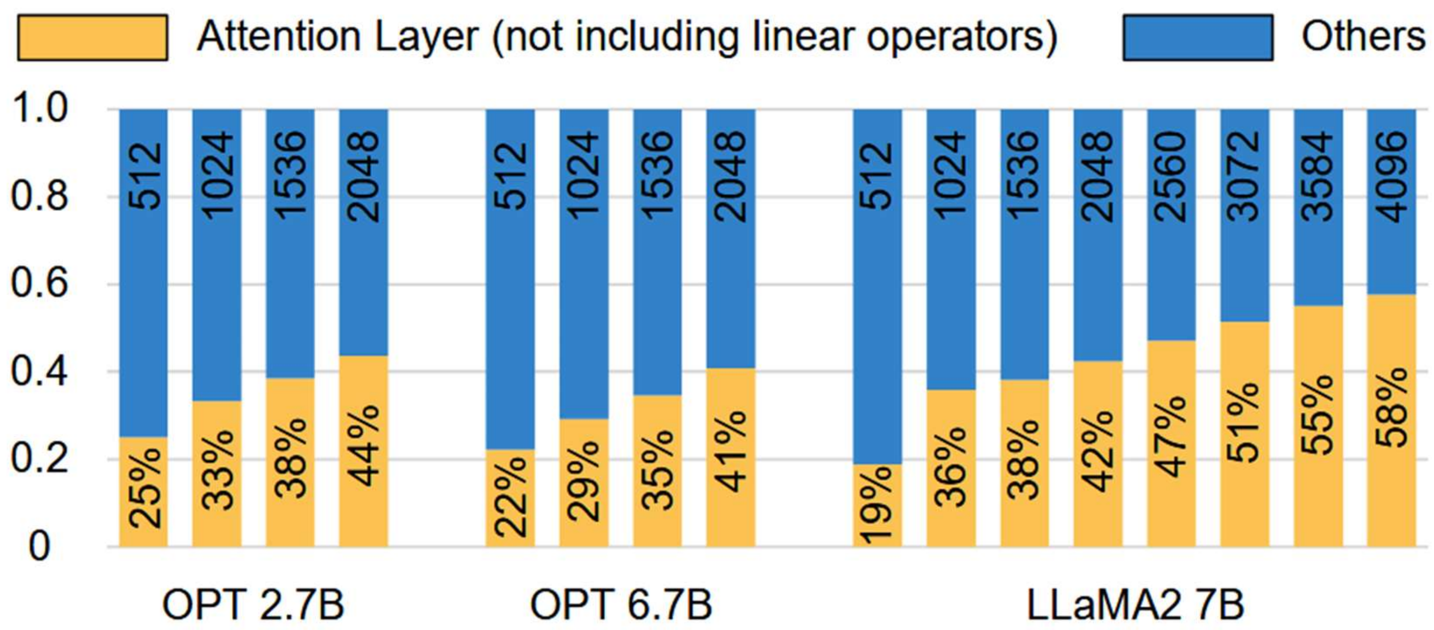
Restored from main memory

Computed in current step

Accumulated load times

	To	be	or	not	to	be
Decode 1	1	1	1	1		
Decode 2	2	2	2	2	1	
Decode 3	3	3	3	3	2	1

The inefficiency of the KV cache



- LLM decoding is bottlenecked by the huge cost of accessing the KV cache
- Attention accounts for over 50% of the end to end latency for long sequence decoding

Motivational ideas

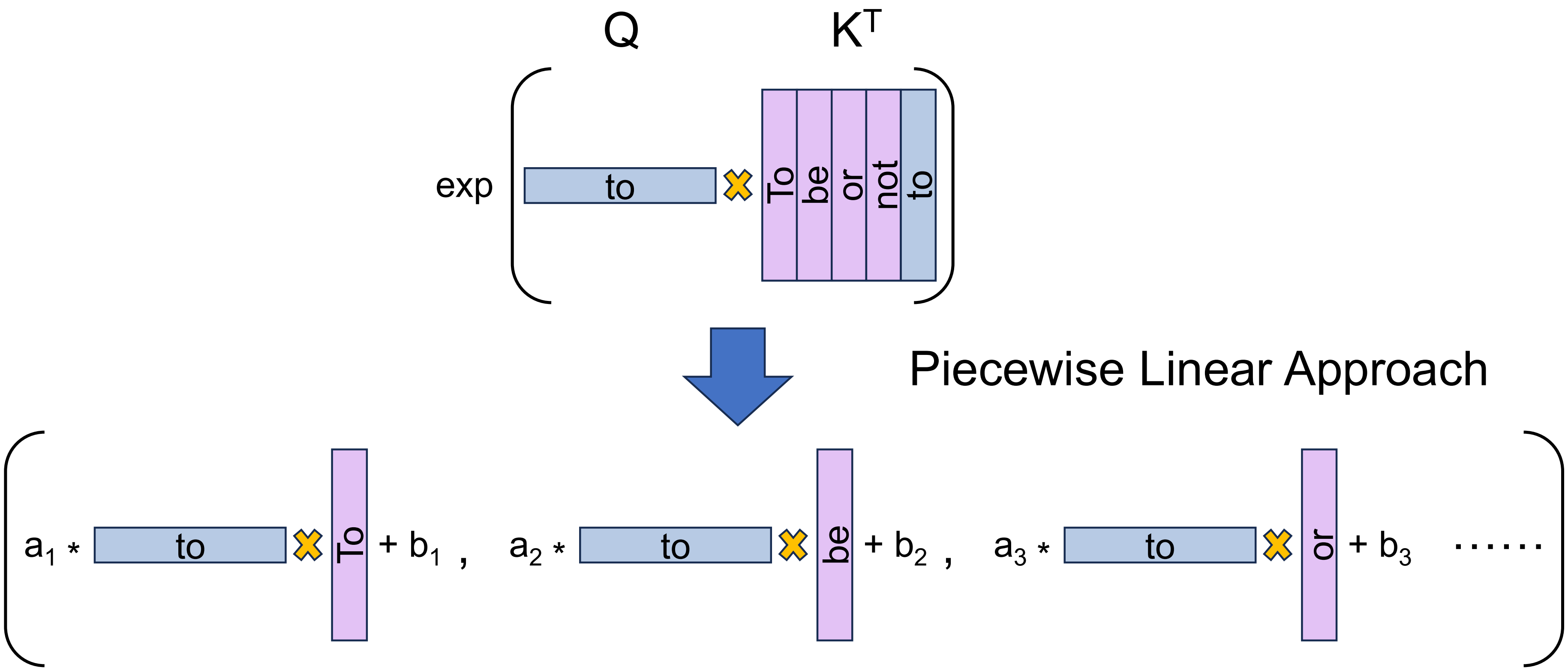
Key Idea

Remembering the complete decoding history is important for the LLM ability

Is it possible to access the **complete history without** retrieving the **entire KV cache**?

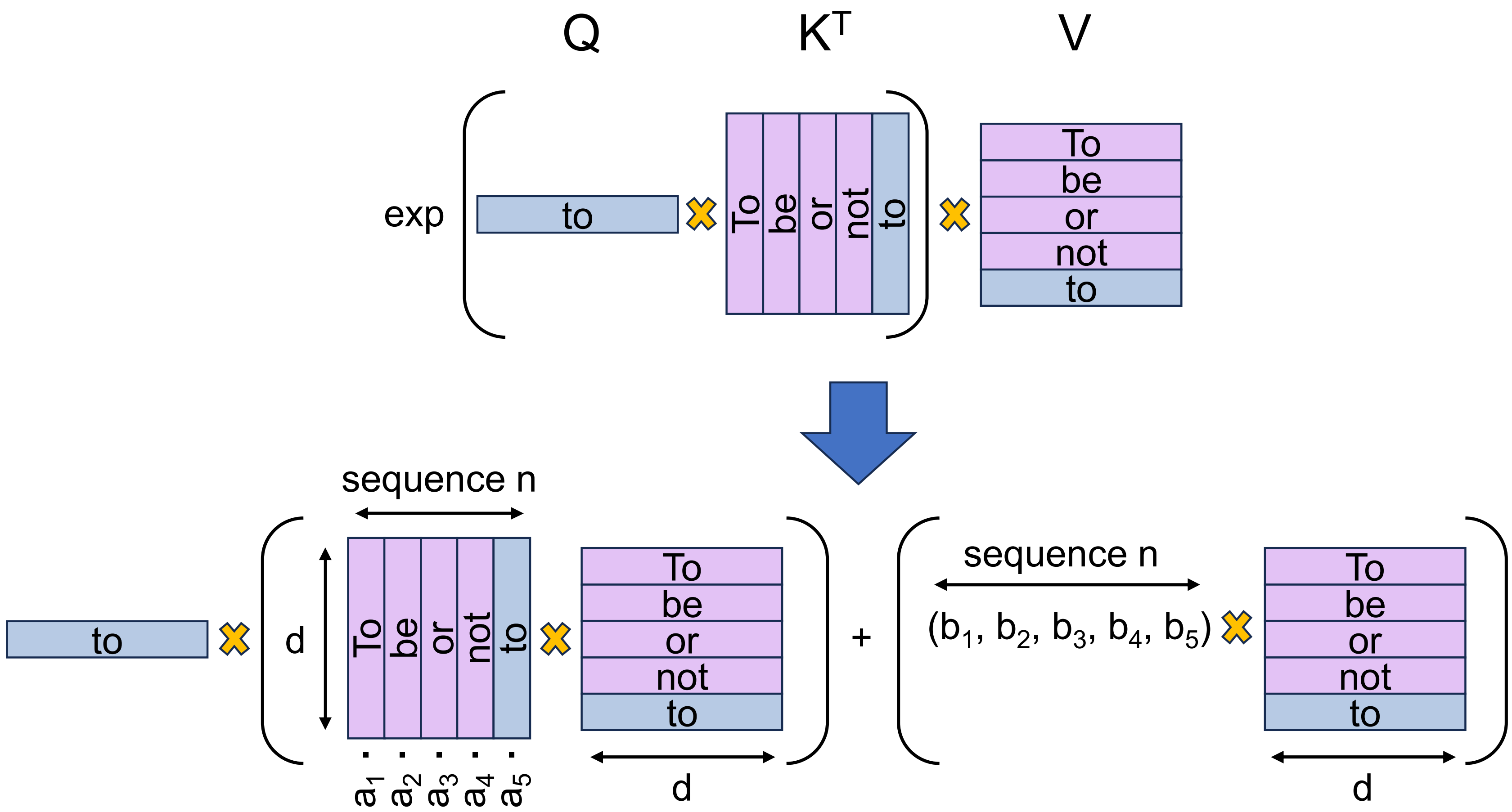
combine K with V into fixed-size intermediate cache, reducing the sequence dimension

Motivational ideas



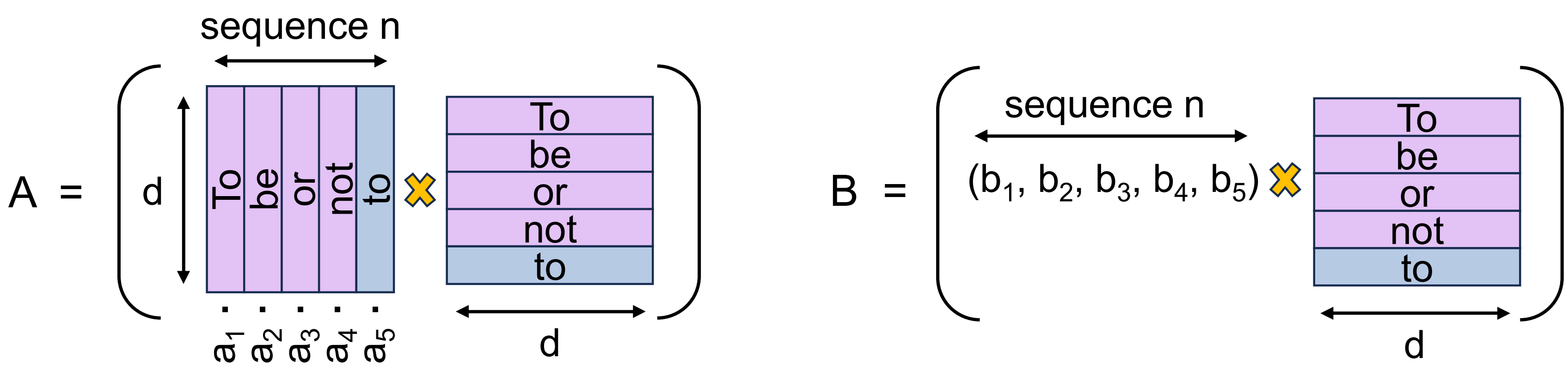
a_i, b_i are linear coefficients determined by the range of the QK_i^T

Motivational ideas



Intermediate caches: d x d tensor 1 x d tensor

Motivational ideas

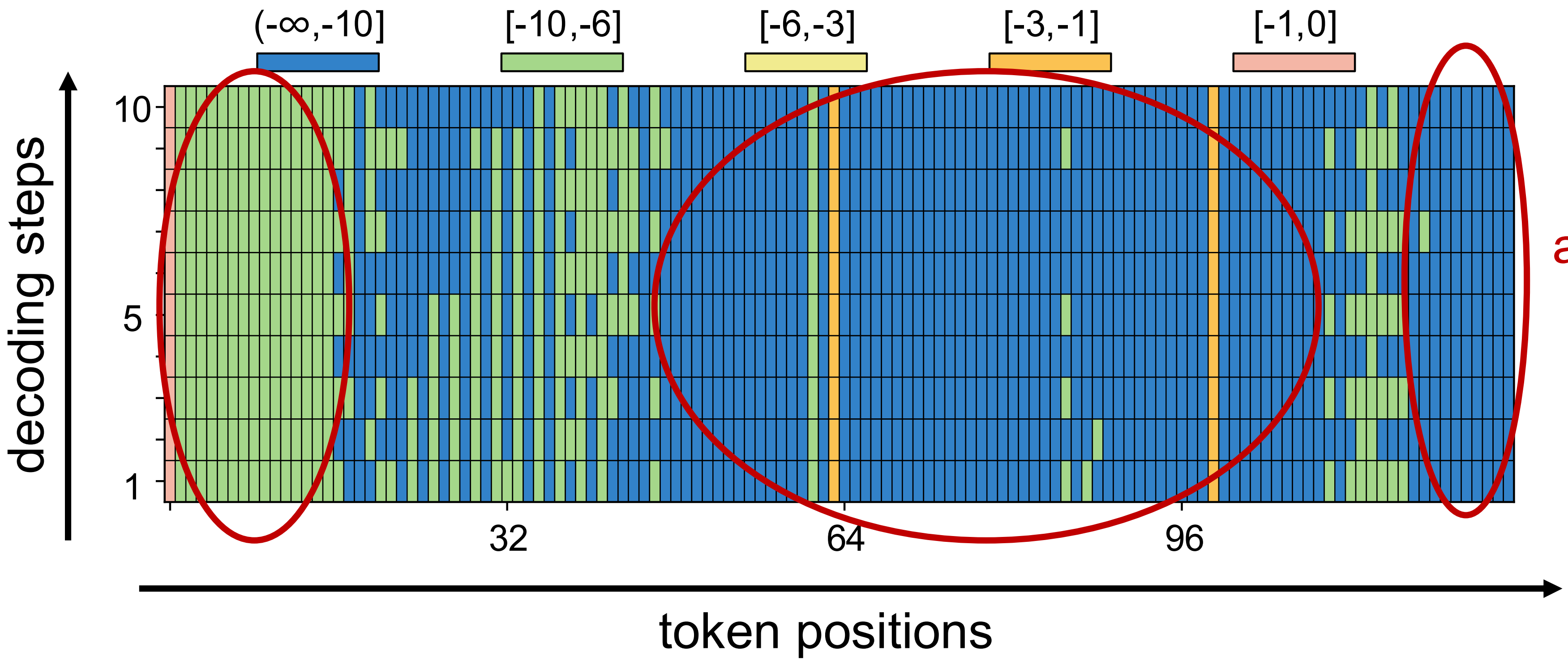


Intermediate caches A and B are fixed-size and much smaller than long context KV cache

Challenge:
How to deal with changing a_i, b_i? Is the cost of maintaining intermediate caches affordable?

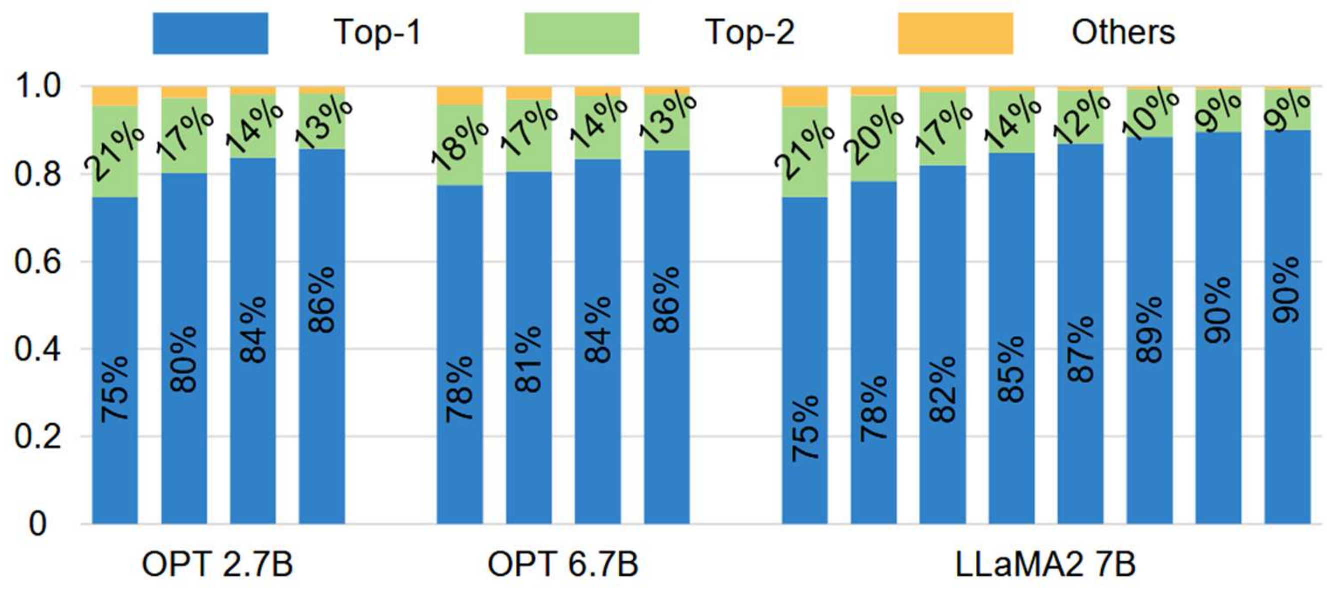
Observation: numerical locality of attention scores

The numerical range of attention scores (QK^T-max) across decoding steps



Most positions' attention scores present good numerical locality

Observation: numerical locality of attention scores

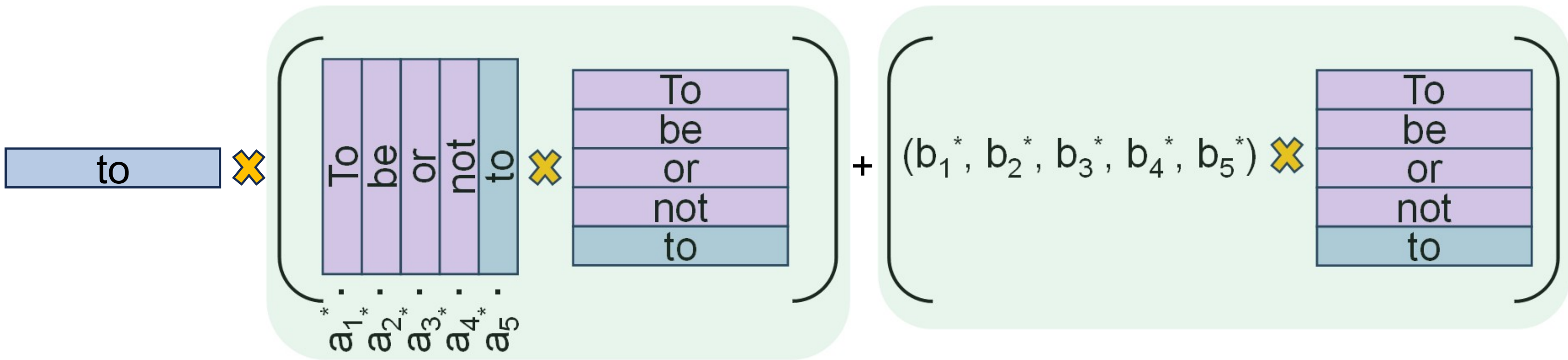


- On average, each position’s attention score has > 74% probability to fall into the its top-1 likely interval
- > 95% probability to fall into its top-1 or top-2 likely intervals

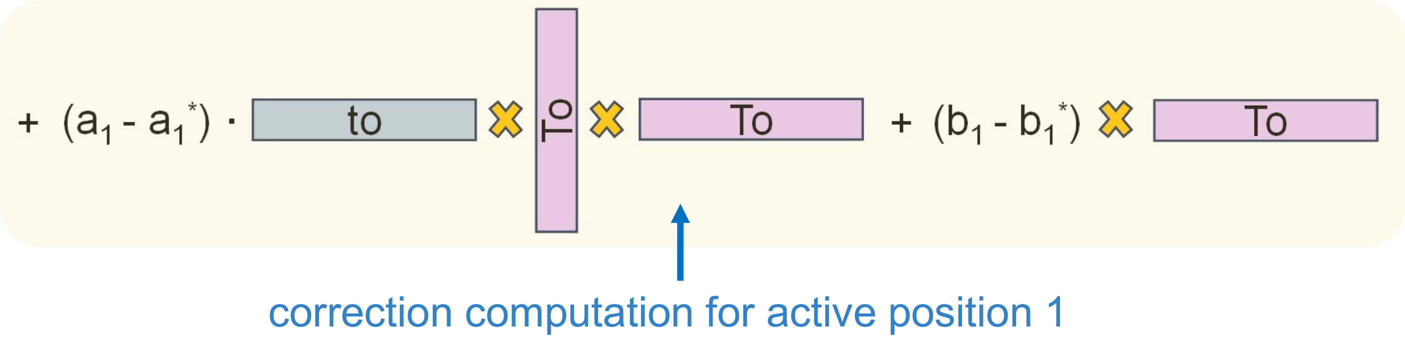
Utilizing locality in attention scores

Key Idea Associate each position i with its mode (i.e. top-1) interval's linear coefficients a_i^*, b_i^* to maintain intermediate caches

Actual interval's coefficients: a_i, b_i Top-1 interval's coefficients: a_i^*, b_i^* suppose $a_1^* \neq a_1$, $b_1^* \neq b_1$



a_i^*, b_i^* rarely change: low cost for updating the intermediate cache



A small portion of positions are active (where a_i, $b_i \neq a_i^*$, b_i^*), thus it's cheap to compute based on a_i^*, b_i^* first and then make corrections

Attention mechanism for decoding

$$o = \frac{\sum_i \left(a_i\left(qk_i^T - m\right) + b_i\right)v_i}{\sum_i \left(a_i\left(qk_i^T - m\right) + b_i\right)}$$

Original attention with piecewise linear method

Attention mechanism for decoding

$$o = \frac{\sum_i \left(a_i\left(qk_i^T - m\right) + b_i\right)v_i}{\sum_i \left(a_i\left(qk_i^T - m\right) + b_i\right)}$$

$$= \frac{qA - mB + C +}{qD - mE + F}$$

Computation based on intermediate caches

Intermediate caches

$$A = \sum_i a_i^* k_i^T v_i \,, B = \sum_i a_i^* v_i \,, C = \sum_i b_i^* v_i$$

$$D = \sum_i a_i^* k_i^T \,, E = \sum_i a_i^* \,, F = \sum_i b_i^*$$

Attention mechanism for decoding

$$o = \frac{\sum_i \left(a_i\left(qk_i^T - m\right) + b_i\right)v_i}{\sum_i \left(a_i\left(qk_i^T - m\right) + b_i\right)}$$

$$= \frac{qA - mB + C + \sum_{i\in J}\alpha_i q k_i^T v_i - m\sum_{i\in J}\alpha_i v_i + \sum_{i\in J}\beta_i v_i}{qD - mE + F + \sum_{i\in J}\alpha_i q k_i^T - m\sum_{i\in J}\alpha_i + \sum_{i\in J}\beta_i}$$

Correction computations for active positions

Coefficient difference

$$\boxed{\alpha_i = a_i - a_i^*, \beta_i = b_i - b_i^*}$$

Efficient attention score computation

Computing accurate attention scores s_i requires accessing the full K cache:

$$s_1 =< q, k_1 >, \dots, s_n =< q, k_n >$$

accessed from main memory

Half the memory access of the original attention mechanism: not efficient

Key Idea $$< q, k_i > \approx \pm < q, k_j > \frac{\|k_i\|}{\|k_j\|} \quad \text{if } \cos(\theta_{k_i, k_j}) \approx \pm 1$$

- Determining attention scores' intervals does not require accurate computation
- Fetch a small portion of keys without affecting LAD's accuracy

Efficient attention score computation

Key Idea $< q, k_i > \approx \pm < q, k_j > \frac{\|k_i\|}{\|k_j\|}$ if $\cos(\theta_{k_i,k_j}) \approx \pm 1$

keys:

k_0

-0.1 1.1

directional centers: k_0

Efficient attention score computation

Key Idea $$< q, k_i > \approx \pm < q, k_j > \frac{\|k_i\|}{\|k_j\|} \quad \text{if } \cos(\theta_{k_i,k_j}) \approx \pm 1$$

keys:

k_0	k_1
-0.1 1.1	-0.2 2.0

directional centers: k_0

$$\cos(\theta_{k_0,k_1}) = 0.999 > \underbrace{0.98}_{\text{predefined threshold}}$$

Not taken as new direction

Efficient attention score computation

Key Idea $$< q, k_i > \approx \pm < q, k_j > \frac{\|k_i\|}{\|k_j\|} \quad \text{if } \cos(\theta_{k_i,k_j}) \approx \pm 1$$

keys:

k_0	k_1	k_2
-0.1 1.1	-0.2 2.0	1.2 1.6

directional centers: k_0, k_2

$$\cos(\theta_{k_0,k_2}) = 0.742 < 0.98$$ Taken as new direction

Efficient attention score computation

Key Idea $< q, k_i > \approx \pm < q, k_j > \frac{\|k_i\|}{\|k_j\|}$ if $\cos(\theta_{k_i,k_j}) \approx \pm 1$

keys:

k_0	k_1	k_2	k_3
-0.1 1.1	-0.2 2.0	1.2 1.6	0.3 -3.4

directional centers: k_0, k_2

$\cos(\theta_{k_0,k_3}) = -0.999$ $\cos(\theta_{k_2,k_3}) = -0.744$ Not taken as new direction

< -0.98

Efficient attention score computation

Key Idea $< q, k_i > \approx \pm < q, k_j > \frac{\|k_i\|}{\|k_j\|}$ if $\cos(\theta_{k_i,k_j}) \approx \pm 1$

keys:

k_0	k_1	k_2	k_3	k_4
-0.1 1.1	-0.2 2.0	1.2 1.6	0.3 -3.4	-0.3 3.2

directional centers: k_0, k_2

$\cos(\theta_{k_0,k_4}) = 0.999$ $\cos(\theta_{k_2,k_4}) = 0.741$ Not taken as new direction

> 0.98

Efficient attention score computation

Key Idea $< q, k_i > \approx \pm < q, k_j > \dfrac{\|k_i\|}{\|k_j\|}$ if $\cos(\theta_{k_i,k_j}) \approx \pm 1$

keys:

k_0	k_1	k_2	k_3	k_4	k_5
-0.1 1.1	-0.2 2.0	1.2 1.6	0.3 -3.4	-0.3 3.2	-1.1 -1.6

directional centers: k_0, k_2

$\cos(\theta_{k_0,k_5}) = -0.769$ $\cos(\theta_{k_2,k_5}) = -0.999$ Not taken as new direction

< -0.98

Efficient attention score computation

Key Idea $< q, k_i > \approx \pm < q, k_j > \dfrac{\|k_i\|}{\|k_j\|}$ if $\cos(\theta_{k_i,k_j}) \approx \pm 1$

keys:

k_0	k_1	k_2	k_3	k_4	k_5	k_6
-0.1 1.1	-0.2 2.0	1.2 1.6	0.3 -3.4	-0.3 3.2	-1.1 -1.6	-0.2 2.1

directional centers: k_0, k_2

$\cos(\theta_{k_0,k_6}) = \underline{0.999}$ $\cos(\theta_{k_2,k_6}) = 0.739$ Not taken as new direction

> 0.98

Efficient attention score computation

Key Idea $$< q, k_i > \approx \pm < q, k_j > \frac{\|k_i\|}{\|k_j\|} \quad \text{if } \cos(\theta_{k_i,k_j}) \approx \pm 1$$

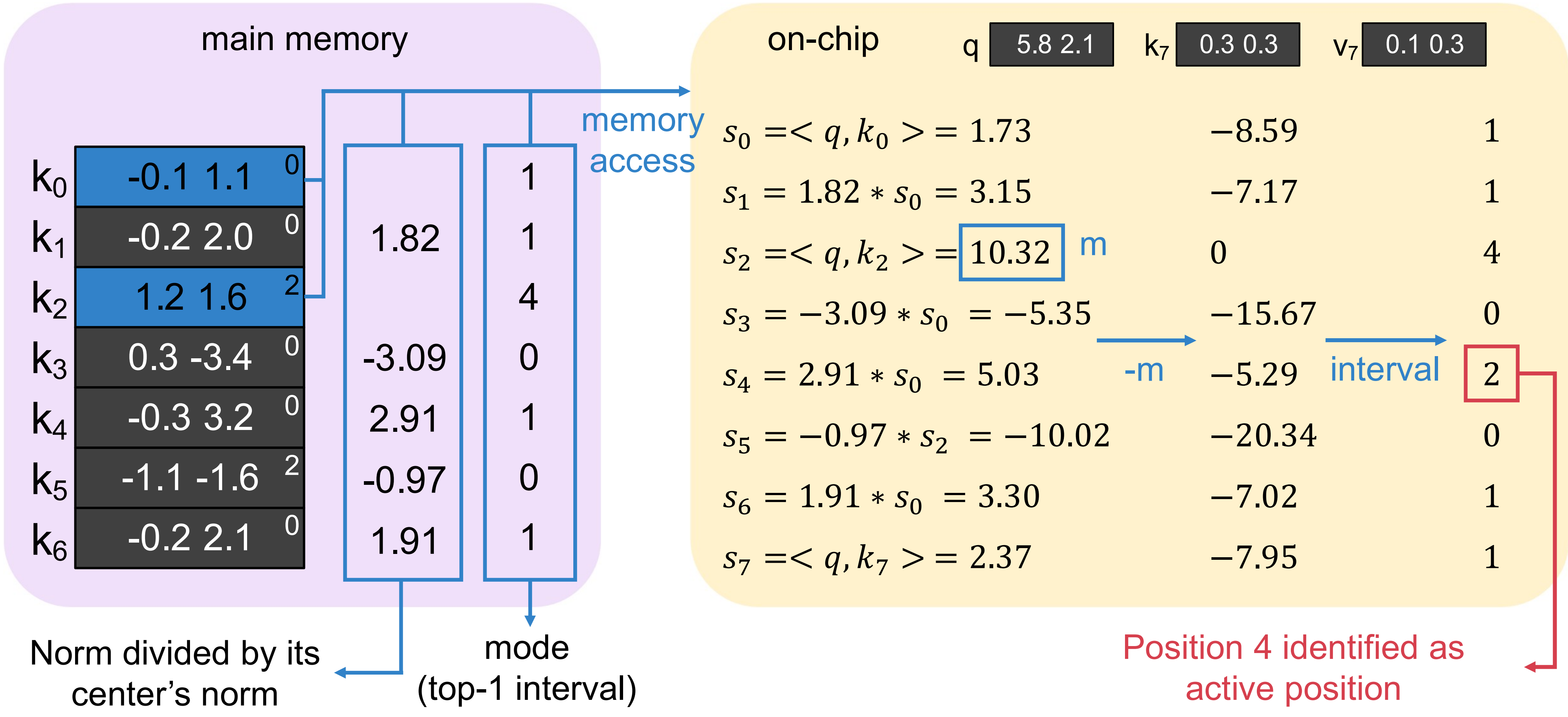
keys:

k_0	k_1	k_2	k_3	k_4	k_5	k_6
-0.1 1.1	-0.2 2.0	1.2 1.6	0.3 -3.4	-0.3 3.2	-1.1 -1.6	-0.2 2.1

Only need to fetch these

- The number of directional centers grows sub-linearly with the keys, more advantageous for long-context scenarios
- Suitable for auto-regressive decoding, where keys are iteratively generated one by one
- Introduce no extra storage of centers

Efficient attention score computation



Compute with mode-based intermediate cache

main memory

$A = \sum_i a_i^* k_i^T v_i$	0.1490 0.0744 0.2012 0.1015	$d \times d$
$B = \sum_i a_i^* v_i$	0.1255 0.0632	d
$C = \sum_i b_i^* v_i$	0.1985 0.1033	d
$D = \sum_i a_i^* k_i$	0.7458 0.9991	d
$E = \sum_i a_i^*$	0.6238	1
$F = \sum_i b_i^*$	0.9610	1

intermediate caches

memory access

on-chip

$$numerator \;=\; qA - mB + C$$
$$= (0.1907, 0.0961)$$

$$denominator \;=\; qD - mE + C$$
$$= 0.9469$$

$$\frac{qA - mB + C + \sum_{i\in J}\left(\alpha_i\left(qk_i^T - m\right) + \beta_i\right)v_i}{qD - mE + F + \sum_{i\in J}\left(\alpha_i\left(qk_i^T - m\right) + \beta_i\right)}$$

Correction for active positions

main memory

	k_0	k_1	k_2	k_3	k_4	k_5	k_6
	-0.1 1.1	-0.2 2.0	1.2 1.6	0.3 -3.4	-0.3 3.2	-1.1 -1.6	-0.2 2.1
	v_0	v_1	v_2	v_3	v_4	v_5	v_6
	0.5 0.4	0.3 0.3	0.2 0.1	0.5 -0.5	0.4 0.2	0.2 0.3	1.0 1.1

only active position’s KV needs to be accessed

on-chip

① compute accurate attention scores for active positions, confirm its actual interval

$s_4 = < q, k_4 > = 4.98$ → $s_4 - m = -5.34$ → in interval 2, while mode interval is 1

$$\frac{qA - mB + C + \sum_{i\in J}\left(\alpha_i\left(qk_i^T - m\right) + \beta_i\right)v_i}{qD - mE + F + \sum_{i\in J}\left(\alpha_i\left(qk_i^T - m\right) + \beta_i\right)}$$

Correction for active positions

main memory

	k_0	k_1	k_2	k_3	k_4	k_5	k_6
k	-0.1 1.1	-0.2 2.0	1.2 1.6	0.3 -3.4	-0.3 3.2	-1.1 -1.6	-0.2 2.1
	v_0	v_1	v_2	v_3	v_4	v_5	v_6
v	0.5 0.4	0.3 0.3	0.2 0.1	0.5 -0.5	0.4 0.2	0.2 0.3	1.0 1.1

only active position's KV needs to be accessed

on-chip

① compute accurate attention scores for active positions, confirm its actual interval

$s_4 = < q, k_4 > = 4.98 \rightarrow s_4 - m = -5.34 \rightarrow$ in interval 2, while mode interval is 1

② compute coefficient differences for active positions

$\alpha_4 = a_2 - a_1 = 0.0133 \quad \beta_4 = b_2 - b_1 = 0.0735$

$$\frac{qA - mB + C + \sum_{i\in J}(\alpha_i(qk_i^T - m) + \beta_i)v_i}{qD - mE + F + \sum_{i\in J}(\alpha_i(qk_i^T - m) + \beta_i)}$$

Correction for active positions

main memory

	k_0	k_1	k_2	k_3	k_4	k_5	k_6
k	-0.1 1.1	-0.2 2.0	1.2 1.6	0.3 -3.4	-0.3 3.2	-1.1 -1.6	-0.2 2.1
	v_0	v_1	v_2	v_3	v_4	v_5	v_6
v	0.5 0.4	0.3 0.3	0.2 0.1	0.5 -0.5	0.4 0.2	0.2 0.3	1.0 1.1

only active position's KV needs to be accessed

on-chip

① compute accurate attention scores for active positions, confirm its actual interval

$s_4 = < q, k_4 > = 4.98 \rightarrow s_4 - m = -5.34 \rightarrow$ in interval 2, while mode interval is 1

② compute coefficient differences for active positions

$\alpha_4 = a_2 - a_1 = 0.0133 \quad \beta_4 = b_2 - b_1 = 0.0735$

③ compute correction factors for active positions

$c_4 = \alpha_4(s_4 - m) + \beta_4 = 0.0025$

$$\frac{qA - mB + C + \sum_{i \in J}\left(\alpha_i\left(qk_i^T - m\right) + \beta_i\right)v_i}{qD - mE + F + \sum_{i \in J}\left(\alpha_i\left(qk_i^T - m\right) + \beta_i\right)}$$

Correction for active positions

main memory

	k_0	k_1	k_2	k_3	k_4	k_5	k_6
k	-0.1 1.1	-0.2 2.0	1.2 1.6	0.3 -3.4	-0.3 3.2	-1.1 -1.6	-0.2 2.1
	v_0	v_1	v_2	v_3	v_4	v_5	v_6
v	0.5 0.4	0.3 0.3	0.2 0.1	0.5 -0.5	0.4 0.2	0.2 0.3	1.0 1.1

only active position's KV needs to be accessed

on-chip

① compute accurate attention scores for active positions, confirm its actual interval

$s_4 = < q, k_4 > = 4.98 \rightarrow s_4 - m = -5.34 \rightarrow$ in interval 2, while mode interval is 1

② compute coefficient differences for active positions

$\alpha_4 = a_2 - a_1 = 0.0133 \quad \beta_4 = b_2 - b_1 = 0.0735$

③ compute correction factors for active positions

$c_4 = \alpha_4(s_4 - m) + \beta_4 = 0.0025$

$$\frac{qA - mB + C + \boxed{\sum_{i \in J}\left(\alpha_i\left(qk_i^T - m\right) + \beta_i\right)v_i}}{qD - mE + F + \boxed{\sum_{i \in J}\left(\alpha_i\left(qk_i^T - m\right) + \beta_i\right)}}$$

④ correct the numerator and denominator

$numerator + c_4 v_4 = (0.1916, 0.0966) \qquad denominator + c_4 = 0.9493$

Missing positions and cache updates

on-chip

Add positions missing from intermediate caches to the result

$s_7 = -7.95$ → in interval 1

$$numerator + (a_1 s_7 + b_1) v_7 = (0.1917, 0.0967)$$

$$denominator + (a_1 s_7 + b_1) = 0.9499$$

$$output = \frac{numerator}{denominator} = (0.2018, 0.1018)$$

Update intermediate caches

For each position i changing its mode
(i must be among the active positions)

$$A \mathrel{+}= \alpha_i k_i^T v_i \quad B \mathrel{+}= \alpha_i v_i \quad C \mathrel{+}= \beta_i v_i$$

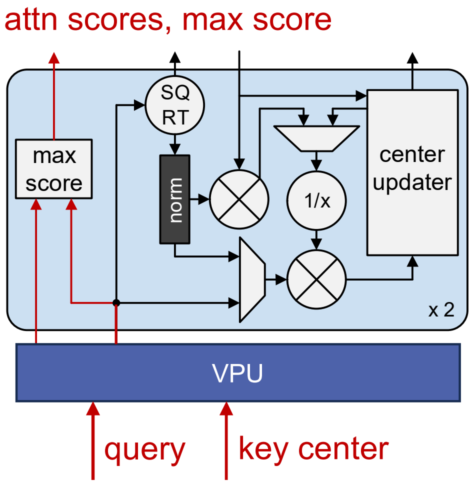
$$D \mathrel{+}= \alpha_i k_i^T \quad E \mathrel{+}= \alpha_i \quad F \mathrel{+}= \beta_i$$

overwrite intermediate caches

main memory

A	B	C	D	E	F
d x d	d	d	d	1	1

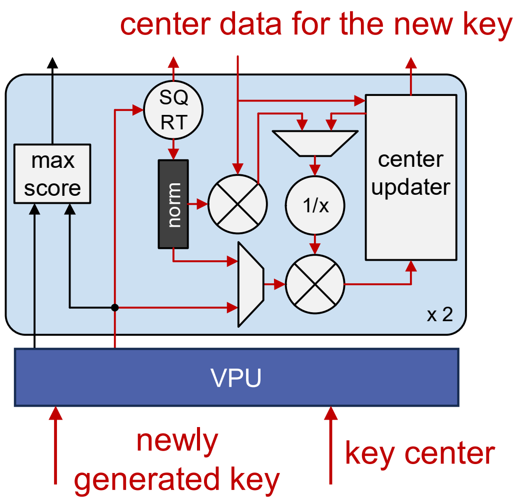
LAD specialized hardware



① Efficient Attention Score (EAS) Module

- Compute approximate attention scores based on key centers
- Update key centers adding the newly generated key

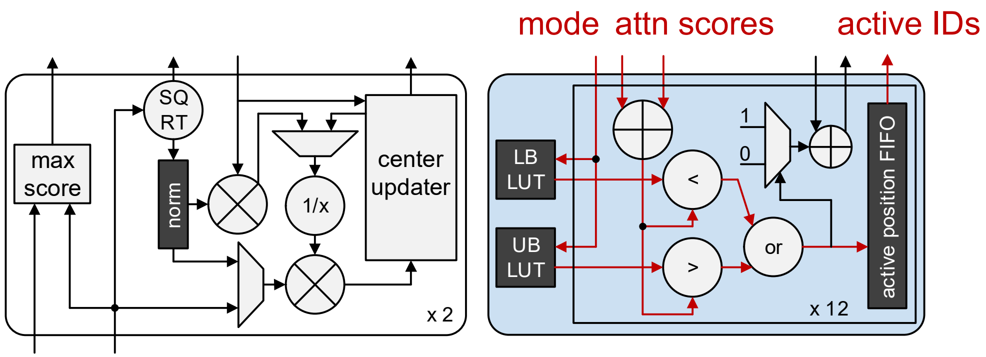
LAD specialized hardware



① Efficient Attention Score (EAS) Module

- Compute approximate attention scores based on key centers
- Update key centers adding the newly generated key

LAD specialized hardware

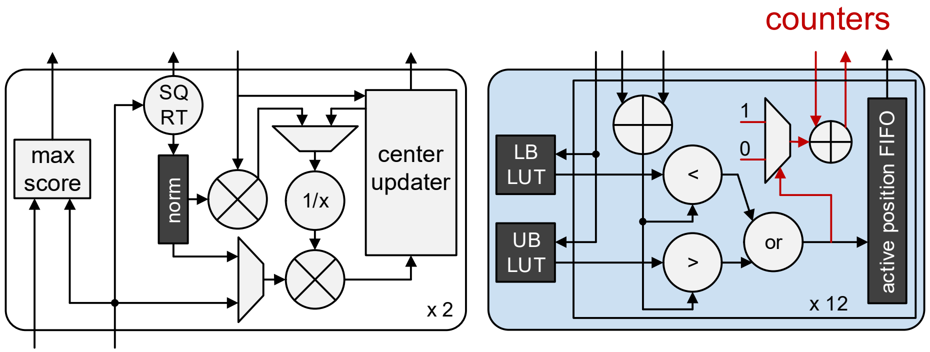


① Efficient Attention Score (EAS) Module

② Active Position Identification (APID) Module

- Compare each position’s score with its mode interval boundary, identify active positions
- Increment non-active positions’ mode interval counters

LAD specialized hardware

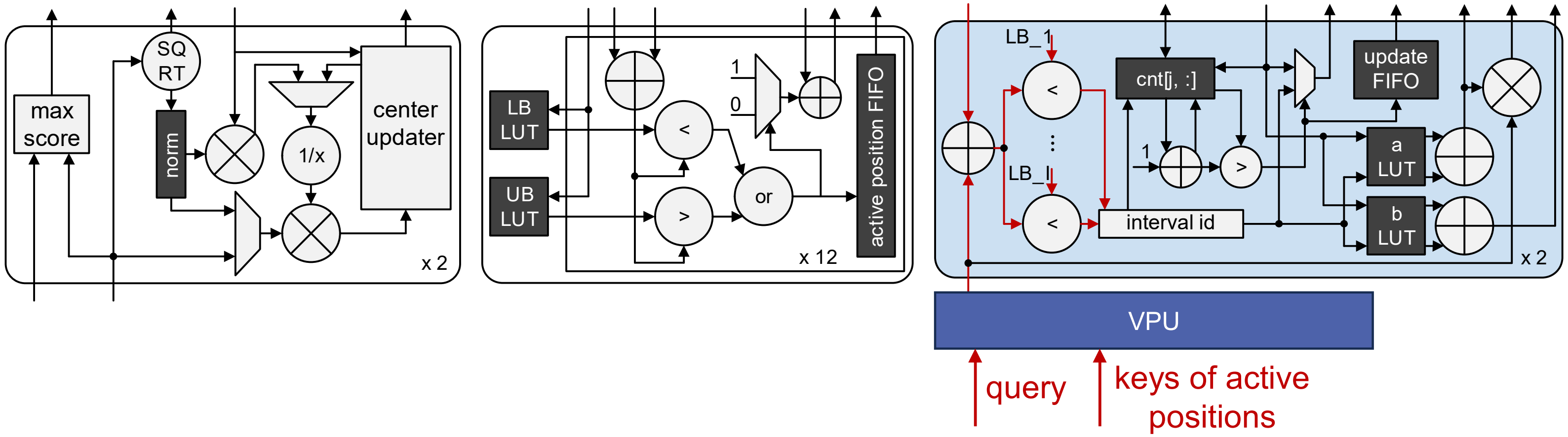


① Efficient Attention Score (EAS) Module

② Active Position Identification (APID) Module

- Compare each position's score with its mode interval boundary, identify active positions
- Increment non-active position's mode interval counter

LAD specialized hardware



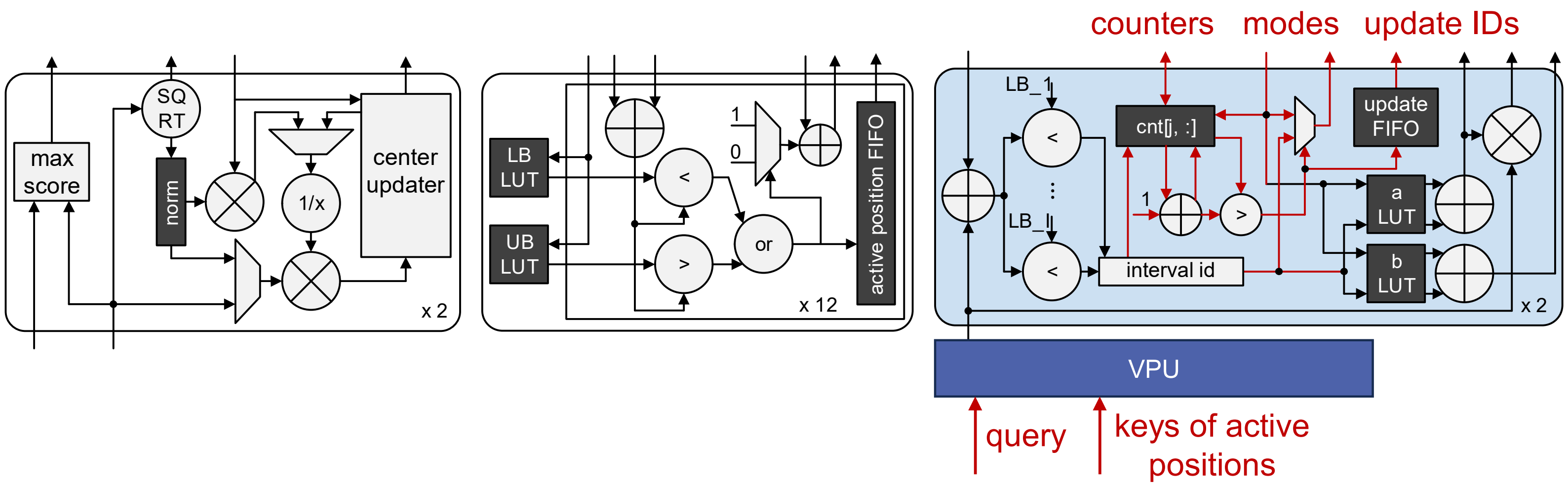
① Efficient Attention Score (EAS) Module

② Active Position Identification (APID) Module

③ Mode Discrepancy (MD) Module

- Compute accurate scores for active positions and confirm their actual intervals
- Increment the actual interval counter and identify updating mode positions
- Compute $\alpha_i, \beta_i, \alpha_i s_i$

LAD specialized hardware



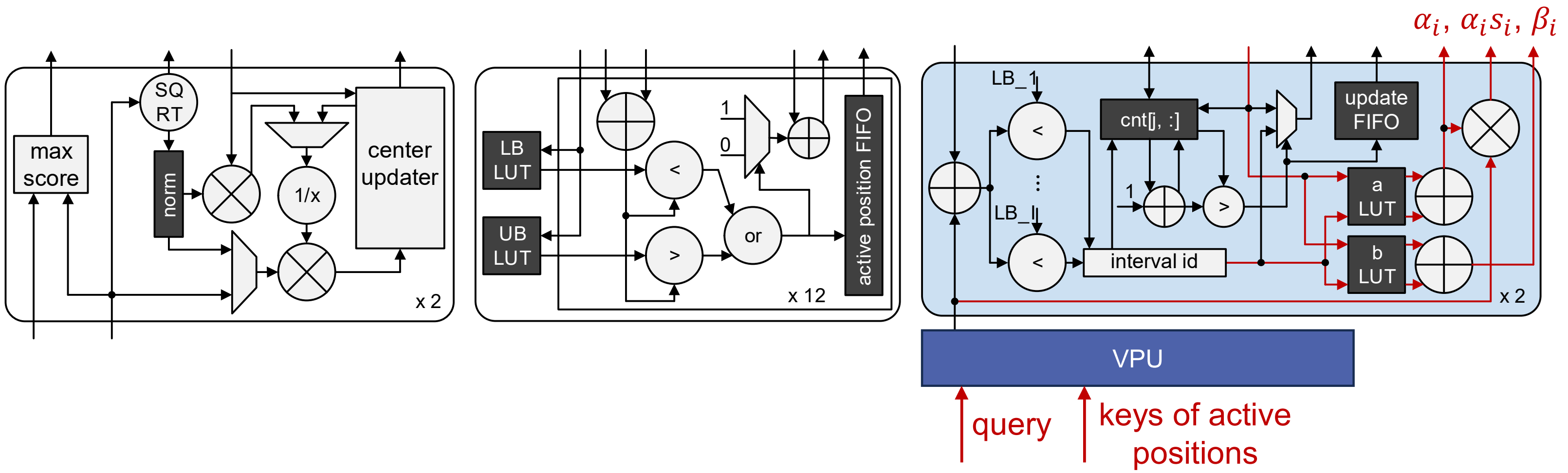
① Efficient Attention Score (EAS) Module

② Active Position Identification (APID) Module

③ Mode Discrepancy (MD) Module

- Compute accurate scores for active positions and confirm their actual intervals
- Increment the actual interval counter and identify updating mode positions
- Compute $\alpha_i, \beta_i, \alpha_i s_i$

LAD specialized hardware



$$\frac{qA - mB + C + \sum_{i\in J}\left(\alpha_i\left(qk_i^T - m\right) + \beta_i\right)v_i}{qD - mE + F + \sum_{i\in J}\left(\alpha_i\left(qk_i^T - m\right) + \beta_i\right)}$$

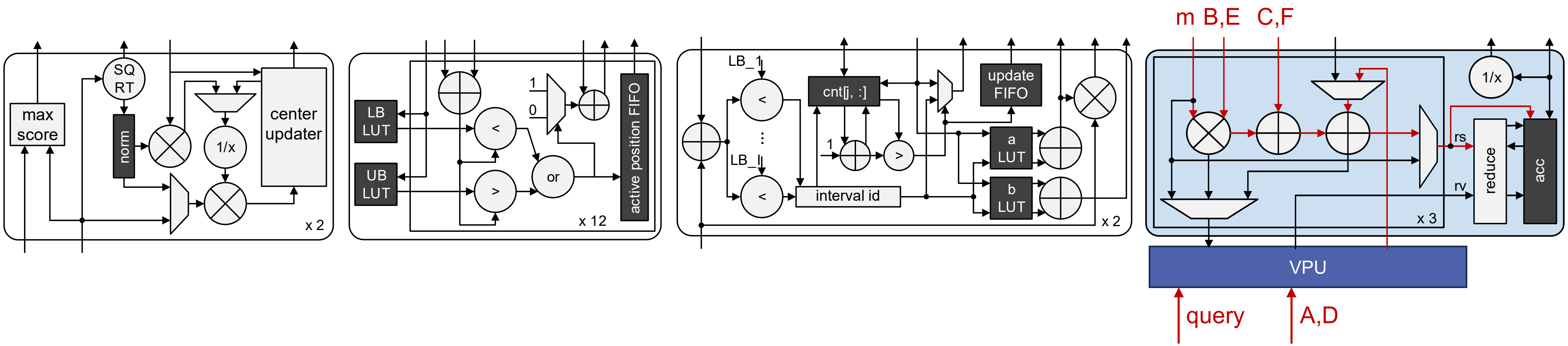
① Efficient Attention Score (EAS) Module

② Active Position Identification (APID) Module

③ Mode Discrepancy (MD) Module

- Compute accurate scores for active positions and confirm their actual intervals
- Increment the actual interval counter and identify updating mode positions
- Compute $\alpha_i,\ \alpha_i s_i,\ \beta_i$

LAD specialized hardware



① Efficient Attention Score (EAS) Module

② Active Position Identification (APID) Module

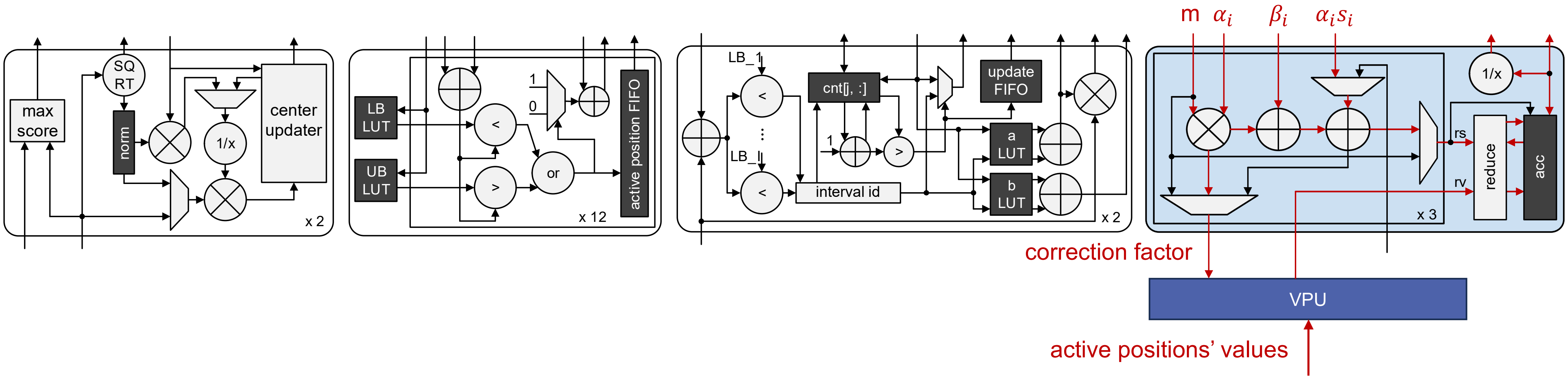
③ Mode Discrepancy (MD) Module

④ Attention Computation (AC) Module

- Compute with intermediate caches
- Compute corrections
- Update intermediate caches

$$\frac{\boxed{qA - mB + C} + \sum_{i \in J}\left(\alpha_i\left(qk_i^T - m\right) + \beta_i\right)v_i}{\boxed{qD - mE + F} + \sum_{i \in J}\left(\alpha_i\left(qk_i^T - m\right) + \beta_i\right)}$$

LAD specialized hardware



① Efficient Attention Score (EAS) Module

② Active Position Identification (APID) Module

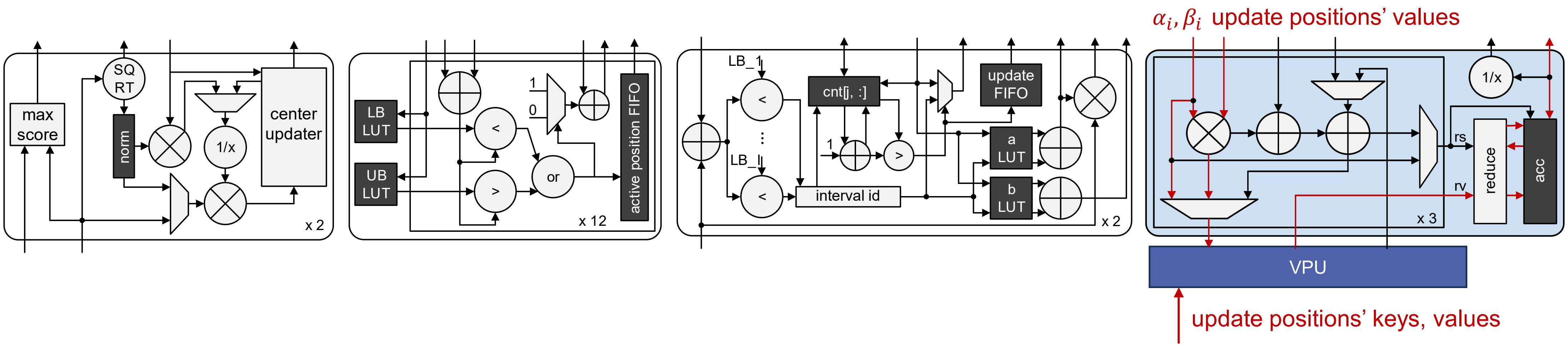
③ Mode Discrepancy (MD) Module

④ Attention Computation (AC) Module

- Compute with intermediate caches
- Compute corrections
- Update intermediate caches

$$\frac{qA - mB + C + \boxed{\sum_{i\in J}\left(\alpha_i\left(qk_i^T - m\right) + \beta_i\right)v_i}}{qD - mE + F + \boxed{\sum_{i\in J}\left(\alpha_i\left(qk_i^T - m\right) + \beta_i\right)}}$$

LAD specialized hardware



① Efficient Attention Score (EAS) Module
② Active Position Identification (APID) Module
③ Mode Discrepancy (MD) Module
④ Attention Computation (AC) Module

- Compute with intermediate caches
- Compute corrections
- Update intermediate caches

Evaluation: accelerator details

Hardware composition

TABLE III
AREA AND POWER OF ONE LAD TILE

Module	Area (mm^2)	Dynamic Power(mW)	Static Power(mW)
Attention Pipeline Modules (not including VPU)			
EAS module	0.003	1.37	0.78
APID module	0.006	2.31	0.99
MD module	0.001	1.06	0.34
AC module	0.087	92.20	20.20
Computation Modules			
VPUs ($\times 7$)	0.398	291.78	77.60
SFM	0.069	43.29	16.90
On-chip SRAM			
SRAM in LAD-1.5 (1.5 MB)	1.596	733.33	118.25
SRAM in LAD-2.5 (2.5 MB)	2.231	841.97	193.58
SRAM in LAD-3.5 (3.5 MB)	3.187	1202.82	276.55
LAD Tile			
LAD-1.5	2.160	1165.34	235.06
LAD-2.5	2.795	1273.98	310.39
LAD-3.5	3.751	1634.83	393.36

- On-chip SRAM accounts for 73-84% of the area and 60-73% power consumption

- Specialized attention pipeline modules account for 17% area and 22% power consumption of on-chip logic excluding SRAM

Evaluation: preserving model accuracy

TABLE I
DECODING ACCURACY EVALUATION: ROUGE SCORES BETWEEN LAD/QSERVE/H2O DECODING RESULTS AND THE ORIGINAL MODEL'S RESULTS

	OPT-2.7B				OPT-6.7B			
	rouge1(%)	rouge2(%)	rougeL(%)	rougeLsum(%)	rouge1(%)	rouge2(%)	rougeL(%)	rougeLsum(%)
alpaca	95.1/NA/22.0	93.8/NA/16.9	94.8/NA/21.5	94.9/NA/21.8	96.7/NA/23.4	95.7/NA/14.0	96.5/NA/22.5	96.5/NA/22.3
gsm8k	98.3/NA/56.5	97.9/NA/48.2	98.2/NA/55.1	98.2/NA/56.1	98.1/NA/53.4	97.6/NA/45.7	98.0/NA/51.4	98.0/NA/52.9
mmlu	97.4/NA/38.3	96.5/NA/28.2	97.2/NA/36.8	97.2/NA/37.6	97.2/NA/39.9	96.2/NA/28.4	96.9/NA/37.9	97.0/NA/38.9
	LLaMA2-7B				LLaMA2-13B			
	rouge1(%)	rouge2(%)	rougeL(%)	rougeLsum(%)	rouge1(%)	rouge2(%)	rougeL(%)	rougeLsum(%)
alpaca	95.9/54.1/19.2	94.4/42.5/17.6	95.4/51.9/19.1	95.5/51.6/19.2	95.8/58.5/19.1	94.0/46.8/17.1	95.3/55.7/19.1	95.6/56.0/19.1
gsm8k	97.2/77.6/54.1	96.4/71.0/49.0	96.9/75.6/52.9	97.0/77.1/53.9	97.2/74.3/56.4	96.2/66.7/47.8	97.0/72.0/53.7	97.1/73.6/55.9
mmlu	96.0/66.3/36.4	94.5/54.7/29.1	95.5/62.5/34.6	95.8/65.0/35.9	95.2/70.6/43.1	93.4/58.9/32.9	94.6/65.9/40.1	95.0/68.6/42.7

LAD generates sequences faithful to the original model: on average 96.3% ROUGE scores between sequences generated by LAD and original models

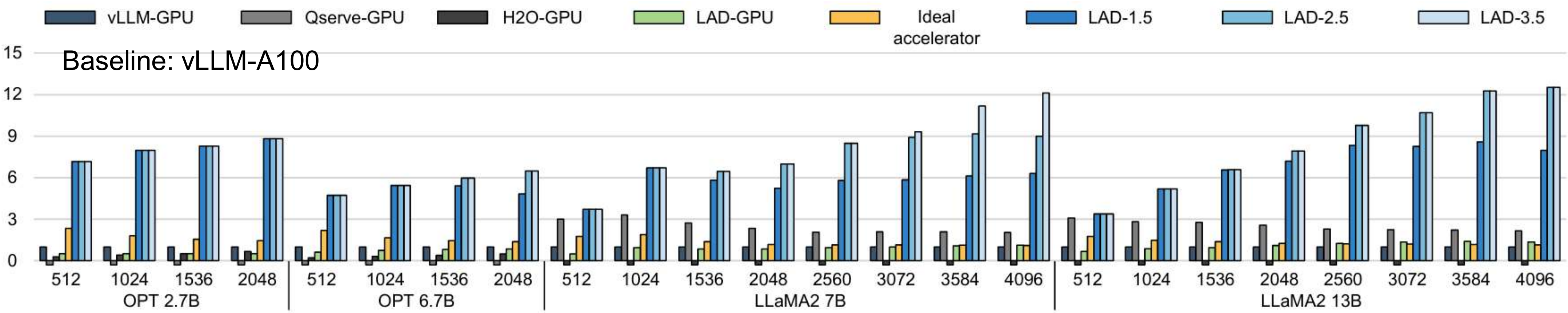
Evaluation: preserving model accuracy

TABLE II
ACCURACY/PERPLEXITY EVALUATION OF ORIGINAL/LAD/QSERVE/H2O MODELS ON POPULAR DATASETS

	OPT-2.7B	OPT-6.7B	LLaMA2-7B	LLaMA2-13B
wikitext2 (ppl)	14.32/14.32/NA/15.72	12.29/12.29/NA/13.38	8.71/8.71/8.83/8.82	7.68/7.68/7.77/7.75
openbookQA (acc)	0.25/0.25/NA/0.16	0.28/0.28/NA/0.15	0.31/0.31/0.31/0.18	0.35/0.35/0.34/0.17
lambada-std (ppl)	7.41/7.40/NA/NA	5.22/5.21/NA/NA	4.13/4.13/4.43/6.43	3.69/3.69/3.78/5.19

No degradation in accuracy metrics was observed across testcases over three popular benchmarks

Evaluation: attention layer performance



(a) Normalized throughput of the attention layer

Scenario 1

Context length of 512-2048

1.5MB SRAM: 5.8x throughput

2.5MB SRAM: 6.2x throughput

3.5MB SRAM: 6.2x throughput

Scenario 2

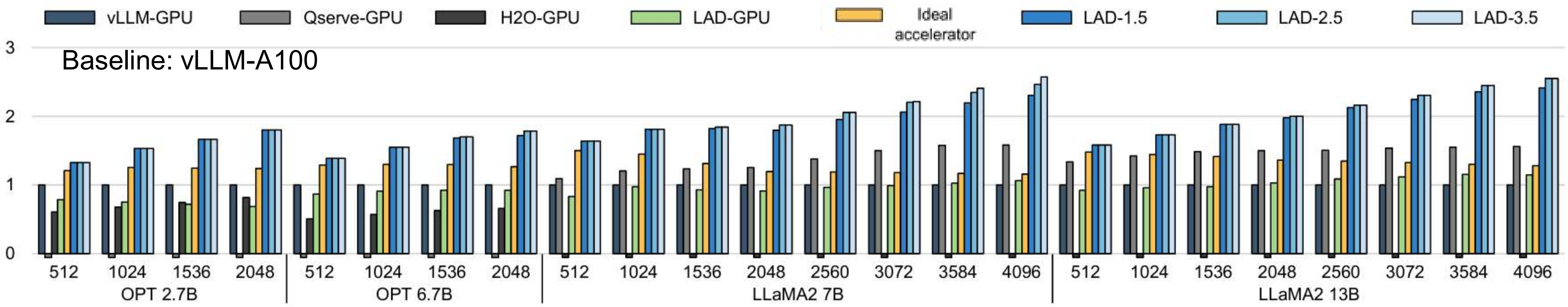
Context length of 2048-4096

1.5MB SRAM: 7.1x throughput

2.5MB SRAM: 10.0x throughput

3.5MB SRAM: 10.7x throughput

Evaluation: end-to-end performance



(b) Normalized throughput of the end-to-end model

Scenario 1

Context length of 512-2048

1.5MB SRAM
1.6x throughput

2.5MB SRAM
1.7x throughput

3.5MB SRAM
1.7x throughput

Scenario 2

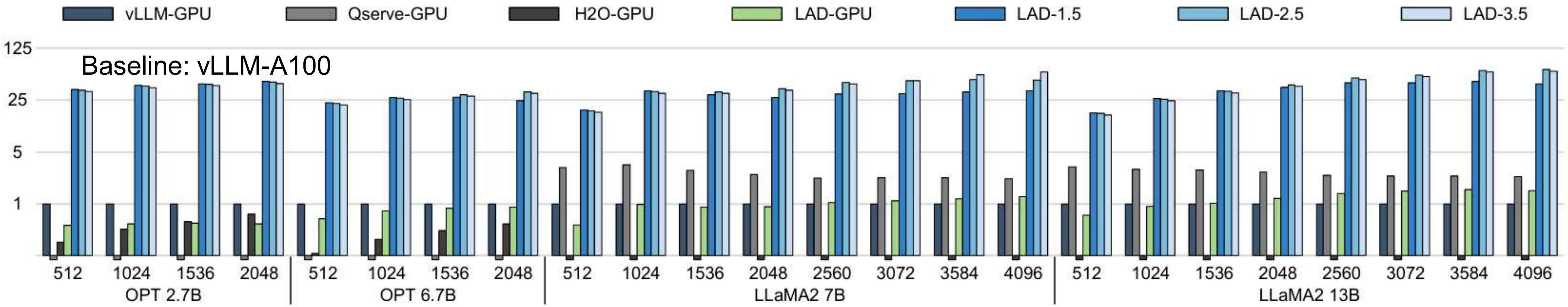
Context length of 2048-4096

1.5MB SRAM
2.2x throughput

2.5MB SRAM
2.3x throughput

3.5MB SRAM
2.3x throughput

Evaluation: attention layer energy efficiency



(a) Normalized energy efficiency of the attention layer

Scenario 1

Context length of 512-2048

1.5MB SRAM

29.3x energy efficiency

2.5MB SRAM

30.4x energy efficiency

3.5MB SRAM

29.0x energy efficiency

Scenario 2

Context length of 2048-4096

1.5MB SRAM

36.9x energy efficiency

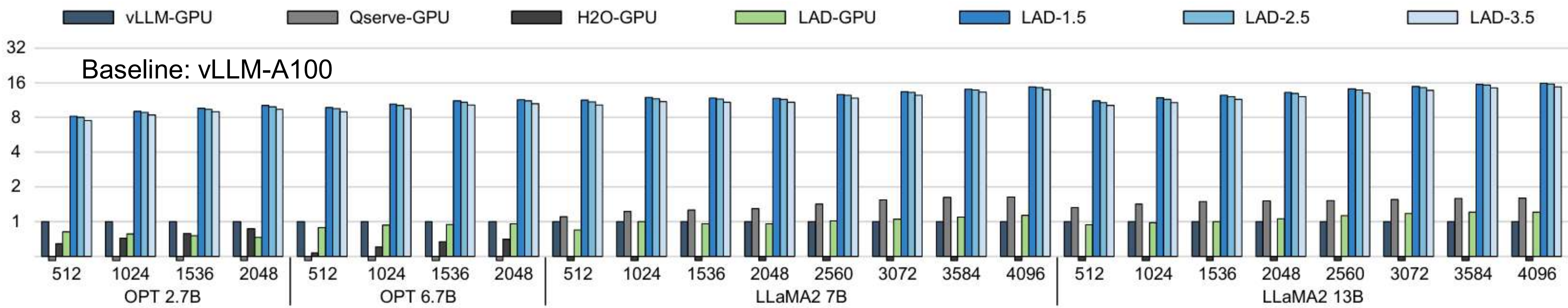
2.5MB SRAM

51.2x energy efficiency

3.5MB SRAM

52.4x energy efficiency

Evaluation: end-to-end energy efficiency



(b) Normalized energy efficiency of the end-to-end model

Scenario 1

Context length of 512-2048

1.5MB SRAM

10.9x energy efficiency

2.5MB SRAM

10.6x energy efficiency

3.5MB SRAM

10.0x energy efficiency

Scenario 2

Context length of 2048-4096

1.5MB SRAM

14.4x energy efficiency

2.5MB SRAM

14.2x energy efficiency

3.5MB SRAM

13.4x energy efficiency

High
Performance
Computer
Architecture

Institute of Computing Technology, CAS

Research Center for Intelligent Computing Systems

Thank you!

Please contact us at the email address below if you have any questions:

wanghaoran20g@ict.ac.cn