DecDEC: A Systems Approach to Advancing Low-Bit LLM Quantization

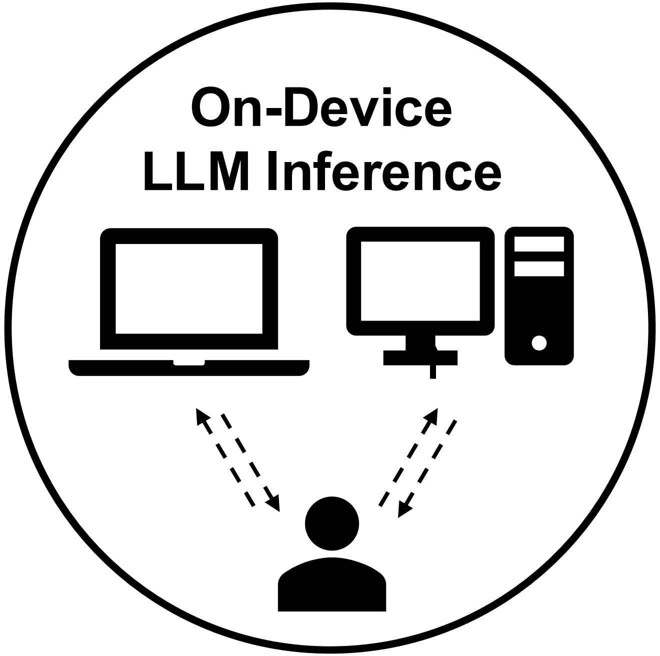
Yeonhong Park*, Jake Hyun*, Hojoon Kim, Jae W. Lee

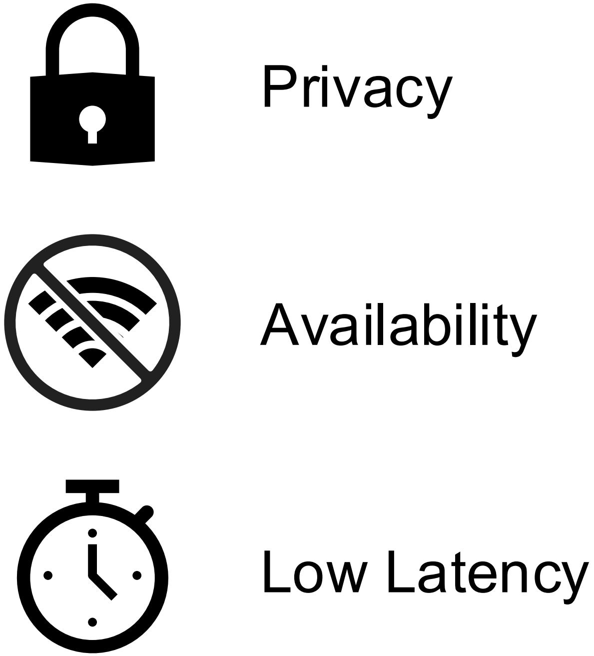
Seoul National University

* These authors equally contributed to this work

Department of CSE
Seoul National University

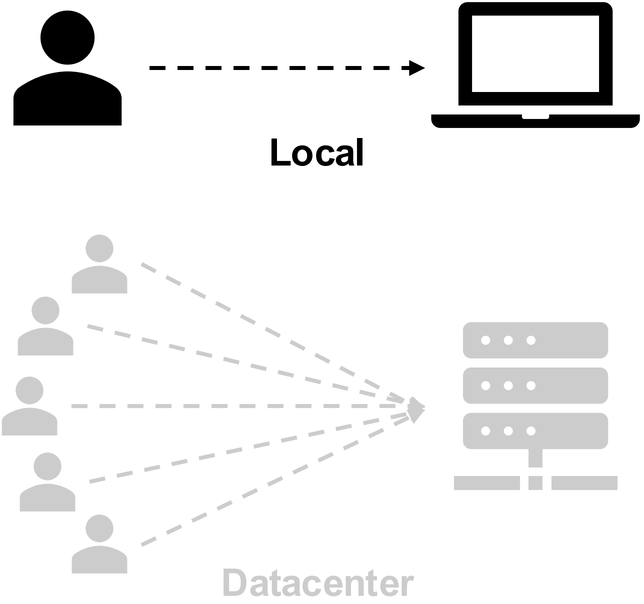
Preliminary: On-Device LLM Inference





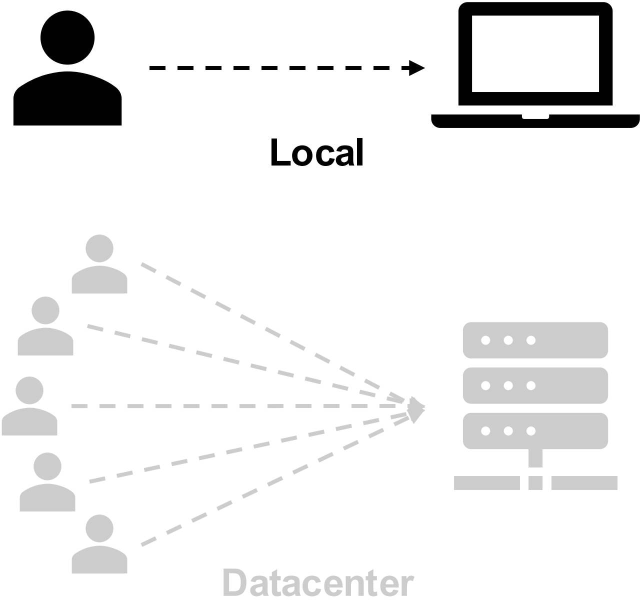
Preliminary: On-Device LLM Inference

Single-Query Inference

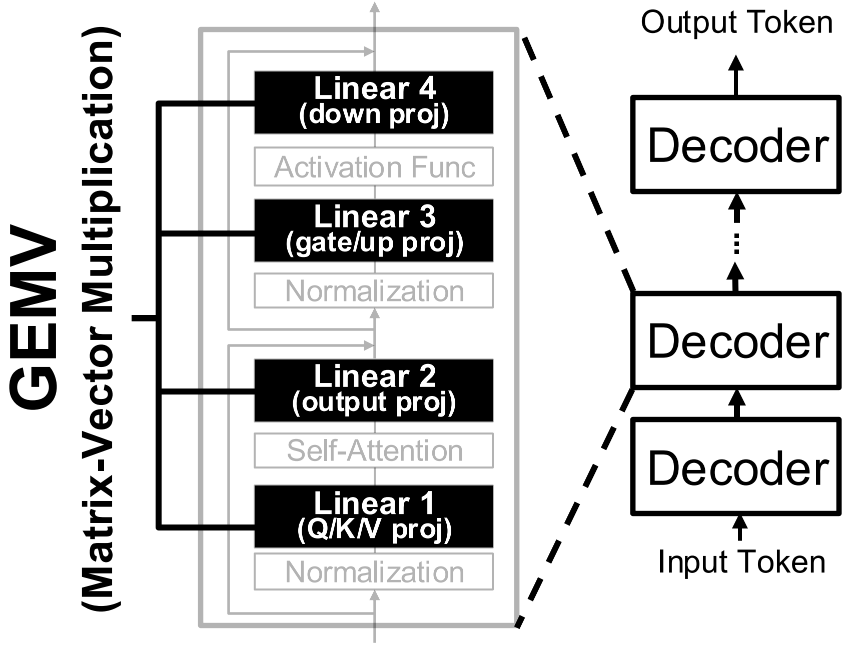


Preliminary: On-Device LLM Inference

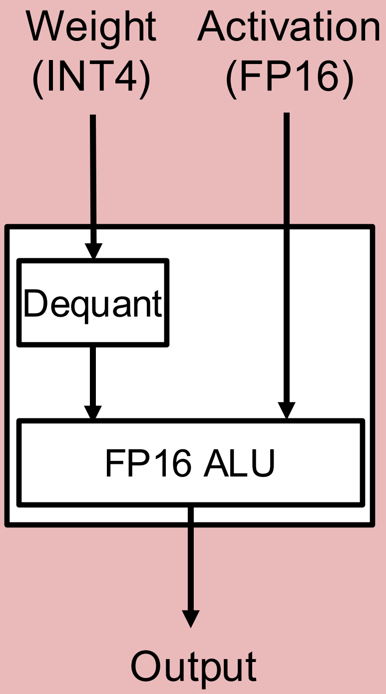
Single-Query Inference



Memory-Bound

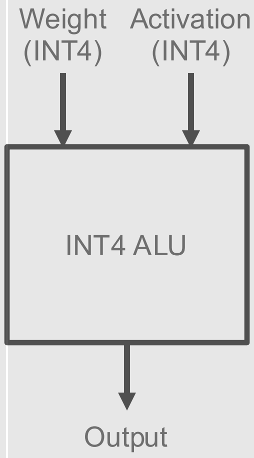


Preliminary: Weight-Only Quantization



Weight-Only Quantization

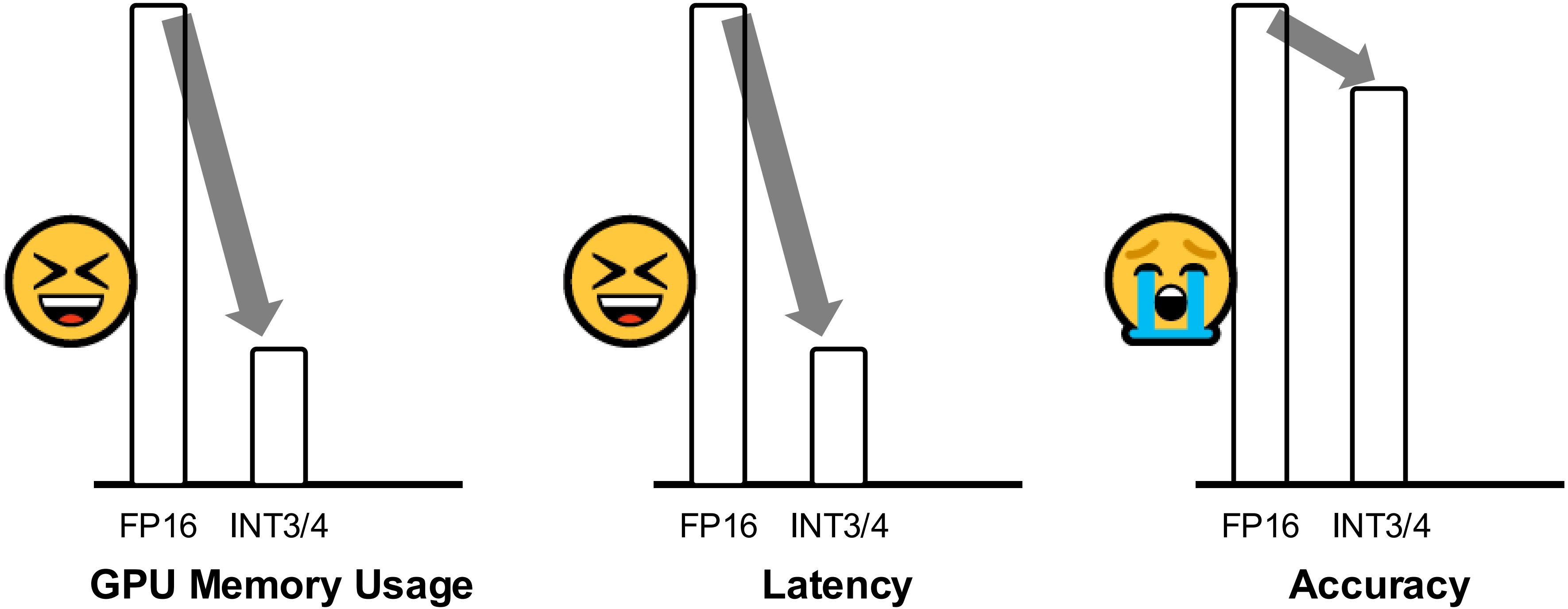
* Save only memory bandwidth
* No loss of activation precision
* Good for single-batch inference (on-device)



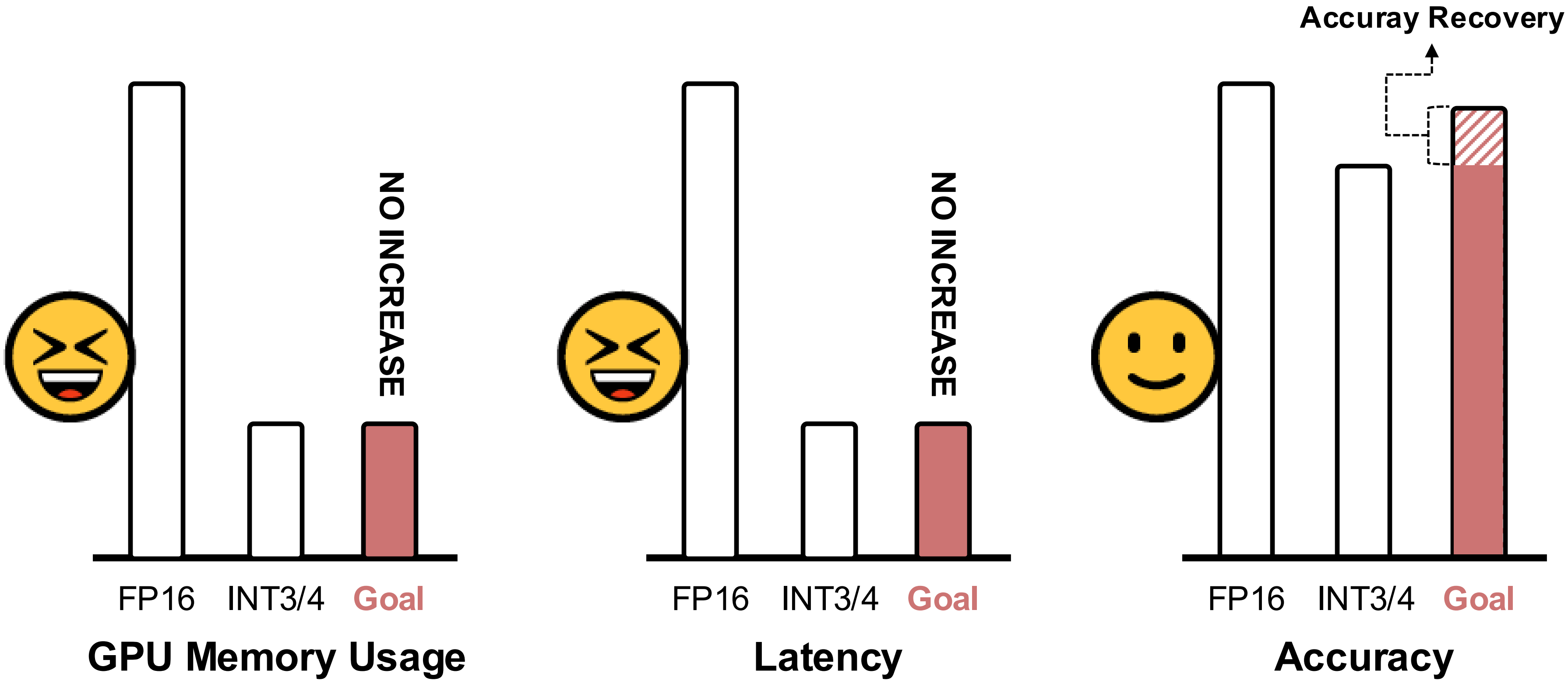
Weight-Activation Quantization

* Save both compute and memory bandwidth
* Good for large-batch inference (data-center)

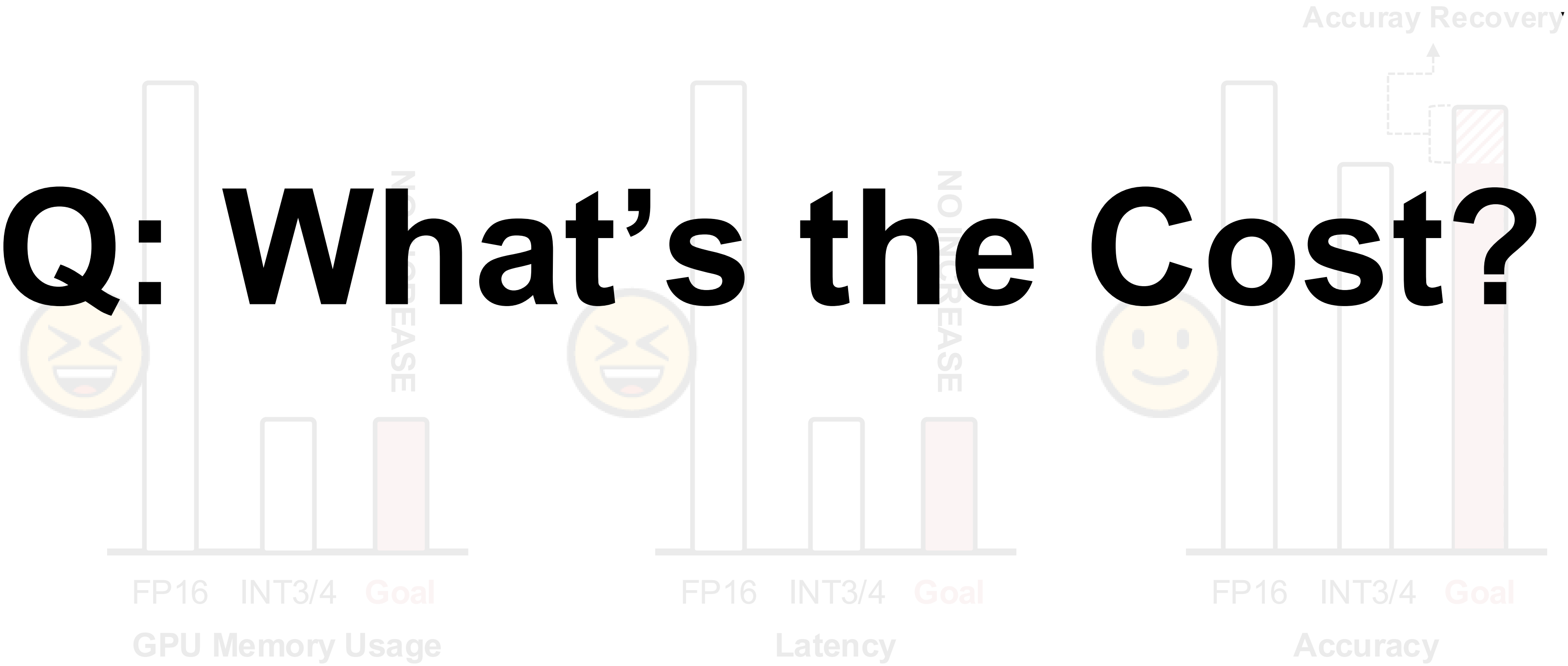
Upsides and Downsides of Quantization



Research Goal: (Almost) Free Accuracy Recovery

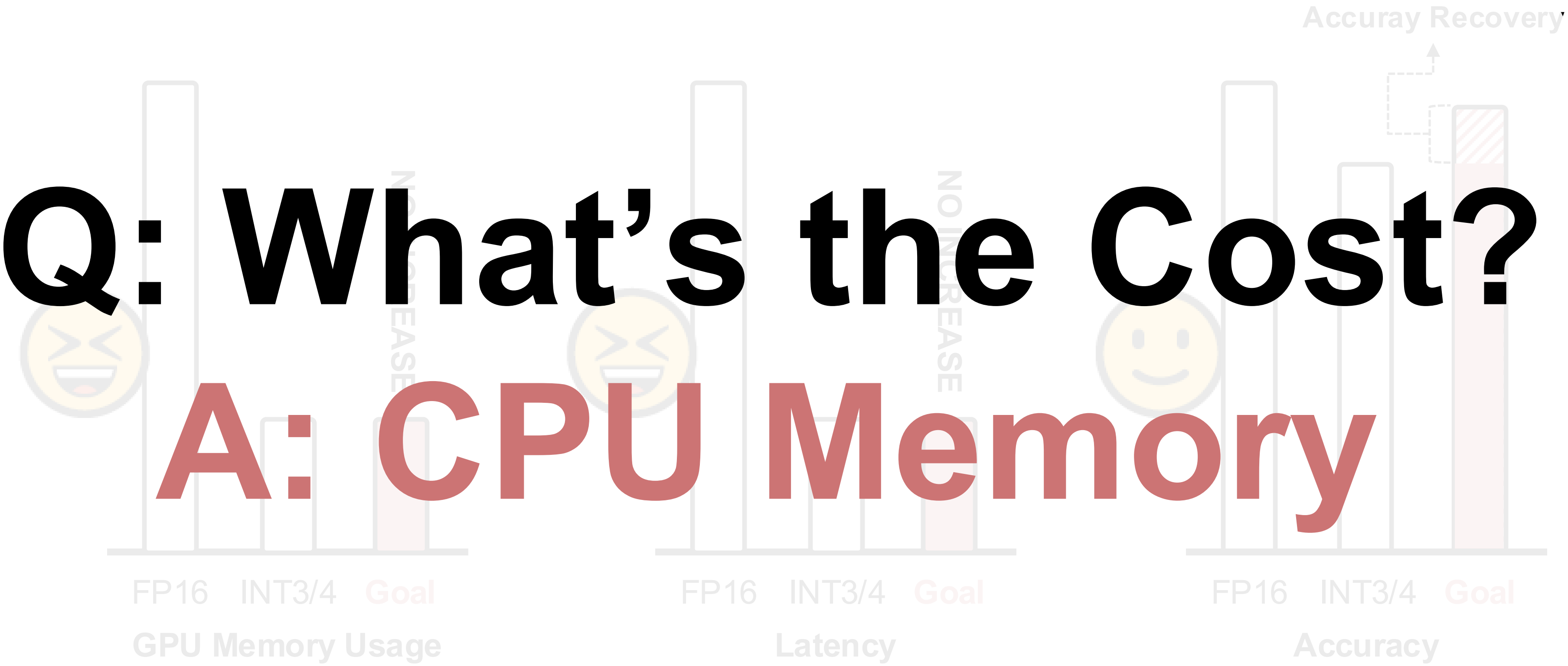


Research Goal: (Almost) Free Accuracy Recovery



Q: What's the Cost?

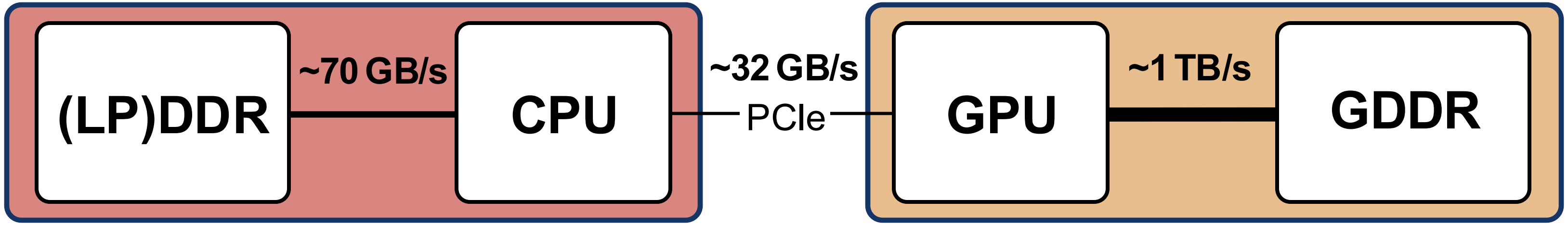
Research Goal: (Almost) Free Accuracy Recovery



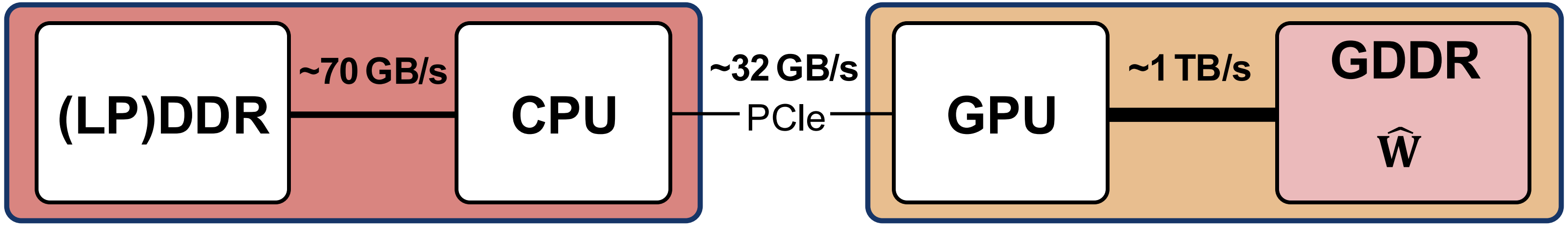
Q: What's the Cost?

A: CPU Memory

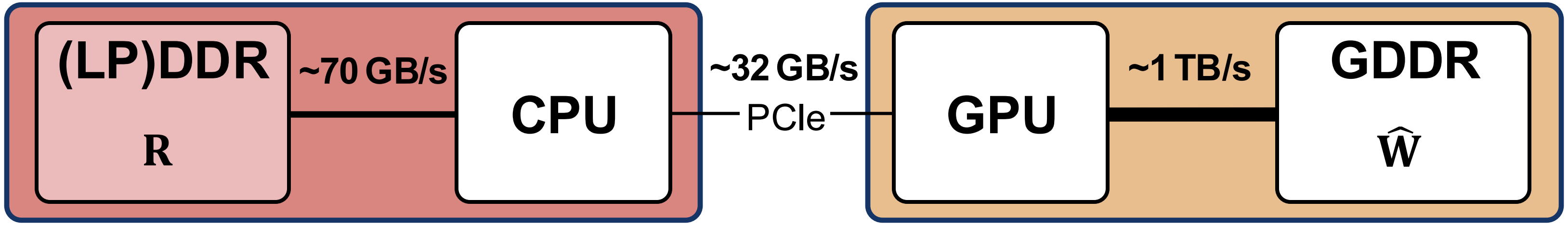
Suggestion: CPU-Augmented Quantized LLM Inference



Suggestion: CPU-Augmented Quantized LLM Inference

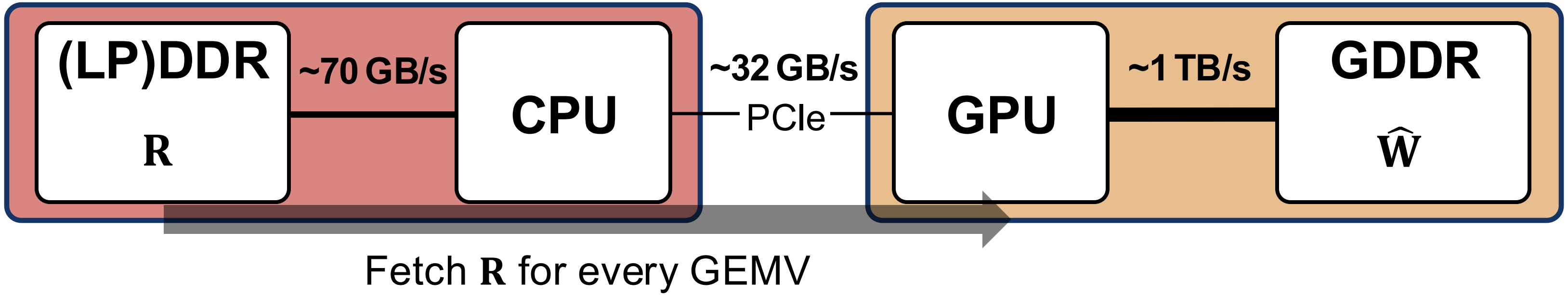


Suggestion: CPU-Augmented Quantized LLM Inference



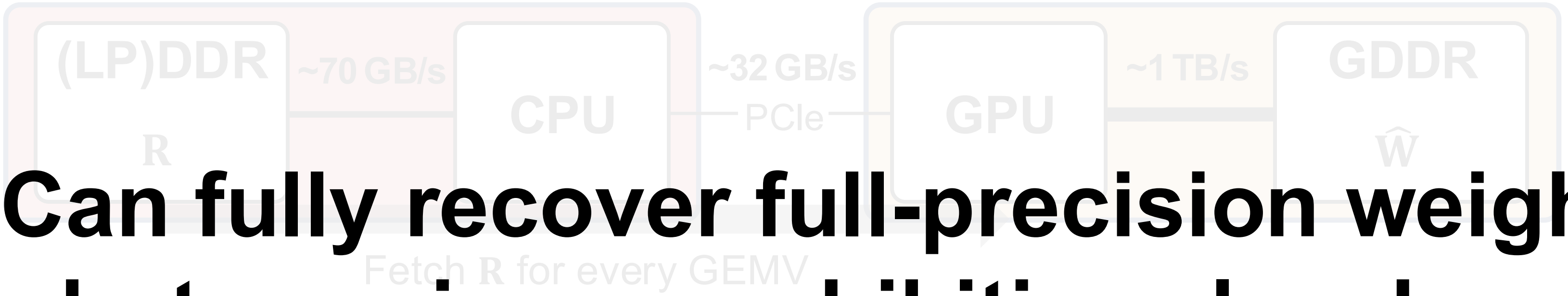
$$\underset{R}{\begin{bmatrix} -0.2 & 0.2 & 0 & 0.1 \\ -0.2 & 0.7 & 0 & 0.3 \\ -0.3 & -0.1 & 0.2 & -0.5 \\ 0 & -0.3 & 0.4 & 0.1 \end{bmatrix}} = \underset{W}{\begin{bmatrix} -3.2 & 0.2 & -1.0 & 3.1 \\ 2.8 & 3.7 & -2.0 & 1.3 \\ 1.7 & -1.1 & 1.2 & 0.5 \\ -4.0 & -2.7 & 0.4 & 2.1 \end{bmatrix}} - \underset{\widehat{W}}{\begin{bmatrix} -3 & 0 & -1 & 3 \\ 3 & 3 & -2 & 1 \\ 2 & -1 & 1 & 1 \\ -4 & -3 & 0 & 2 \end{bmatrix}}$$

Suggestion: CPU-Augmented Quantized LLM Inference



$$(\widehat{\mathbf{W}} + \mathbf{R}) * \mathbf{x} = \mathbf{W} * \mathbf{x}$$

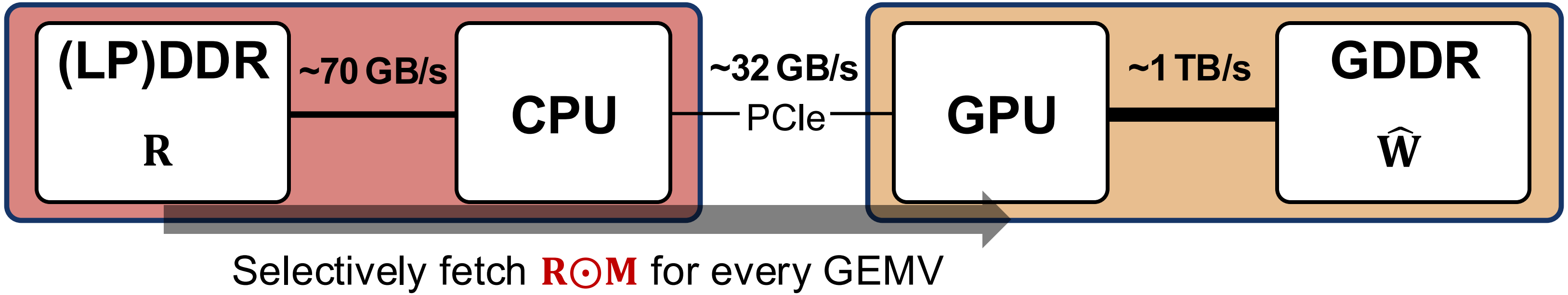
Suggestion: CPU-Augmented Quantized LLM Inference



Can fully recover full-precision weight, but may incur prohibitive slowdown

$$(\widehat{\mathbf{W}} + \mathbf{R}) * \mathbf{x} = \mathbf{W} * \mathbf{x}$$

Suggestion: CPU-Augmented Quantized LLM Inference



$$(\hat{\mathbf{W}} + \mathbf{R} \odot \mathbf{M}) * \mathbf{x} = \mathbf{W} * \mathbf{x}$$

$\mathbf{M}$: binary mask

Key Research Question

How to determine a subset of residuals to fetch ($\mathbf{M}$)?

Key Research Question

How to determine a subset of residuals to fetch ($\mathbf{M}$)?

A good $\mathbf{M}$ should:

* Select the most impactful portions

For RTX-4090, PCIe BW : GPU BW = 32 GB/s : 1 TB/s

$\cong$ **1: 30**

Key Research Question

How to determine a subset of residuals to fetch ($\mathbf{M}$)?

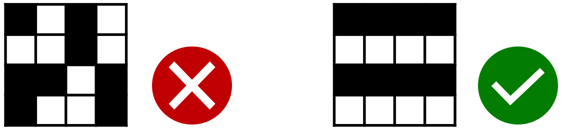
A good $\mathbf{M}$ should:

* Select the most impactful portions

For RTX-4090, PCIe BW : GPU BW = 32 GB/s : 1 TB/s

$\cong$ **1: 30**

* Be structured for efficient processing



Opportunity: Not All Channels are Equally Important

1.1	-0.2	-3.0	0.7

x

*

-3	0	-1	3
3	3	-2	1
2	-1	1	1
-4	-3	0	2

$\hat{W}$

-3.2	0.2	-1.0	3.1
2.8	3.7	-2.0	1.3
1.7	-1.1	1.2	0.5
-4.0	-2.7	0.4	2.1

W

Opportunity: Not All Channels are Equally Important

$$
\underset{x}{\begin{bmatrix} 1.1 & -0.2 & -3.0 & 0.7 \end{bmatrix}} \; * \; \underset{\widehat{W}}{\begin{bmatrix} -3 & 0 & -1 & 3 \\ 3 & 3 & -2 & 1 \\ 2 & -1 & 1 & 1 \\ -4 & -3 & 0 & 2 \end{bmatrix}} \;\longleftarrow\; \underset{W}{\begin{bmatrix} -3.2 & 0.2 & -1.0 & 3.1 \\ 2.8 & 3.7 & -2.0 & 1.3 \\ 1.7 & -1.1 & 1.2 & 0.5 \\ -4.0 & -2.7 & 0.4 & 2.1 \end{bmatrix}}
$$

Opportunity: Not All Channels are Equally Important

$$
\underbrace{\begin{bmatrix} 1.1 & -0.2 & -3.0 & 0.7 \end{bmatrix}}_{\mathrm{X}}
*
\underbrace{\begin{bmatrix} -3 & 0 & -1 & 3 \\ 3 & 3 & -2 & 1 \\ 2 & -1 & 1 & 1 \\ -4 & -3 & 0 & 2 \end{bmatrix}}_{\widehat{\mathrm{W}}}
\leftarrow
\underbrace{\begin{bmatrix} -3.2 & 0.2 & -1.0 & 3.1 \\ 2.8 & 3.7 & -2.0 & 1.3 \\ 1.7 & -1.1 & 1.2 & 0.5 \\ -4.0 & -2.7 & 0.4 & 2.1 \end{bmatrix}}_{\mathrm{W}}
$$

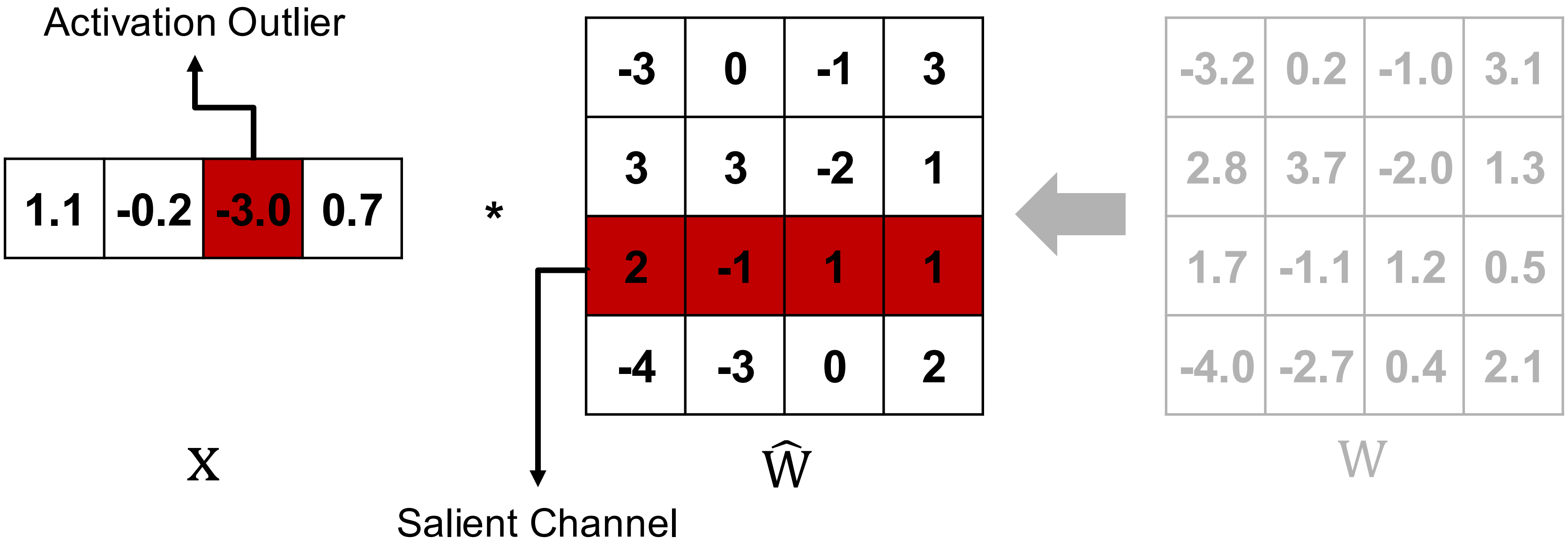
Opportunity: Not All Channels are Equally Important

$$\underset{X}{\begin{bmatrix}1.1 & -0.2 & -3.0 & 0.7\end{bmatrix}} \; * \; \underset{\widehat{W}}{\begin{bmatrix}-3 & 0 & -1 & 3\\ 3 & 3 & -2 & 1\\ 2 & -1 & 1 & 1\\ -4 & -3 & 0 & 2\end{bmatrix}} \;\longleftarrow\; \underset{W}{\begin{bmatrix}-3.2 & 0.2 & -1.0 & 3.1\\ 2.8 & 3.7 & -2.0 & 1.3\\ 1.7 & -1.1 & 1.2 & 0.5\\ -4.0 & -2.7 & 0.4 & 2.1\end{bmatrix}}$$

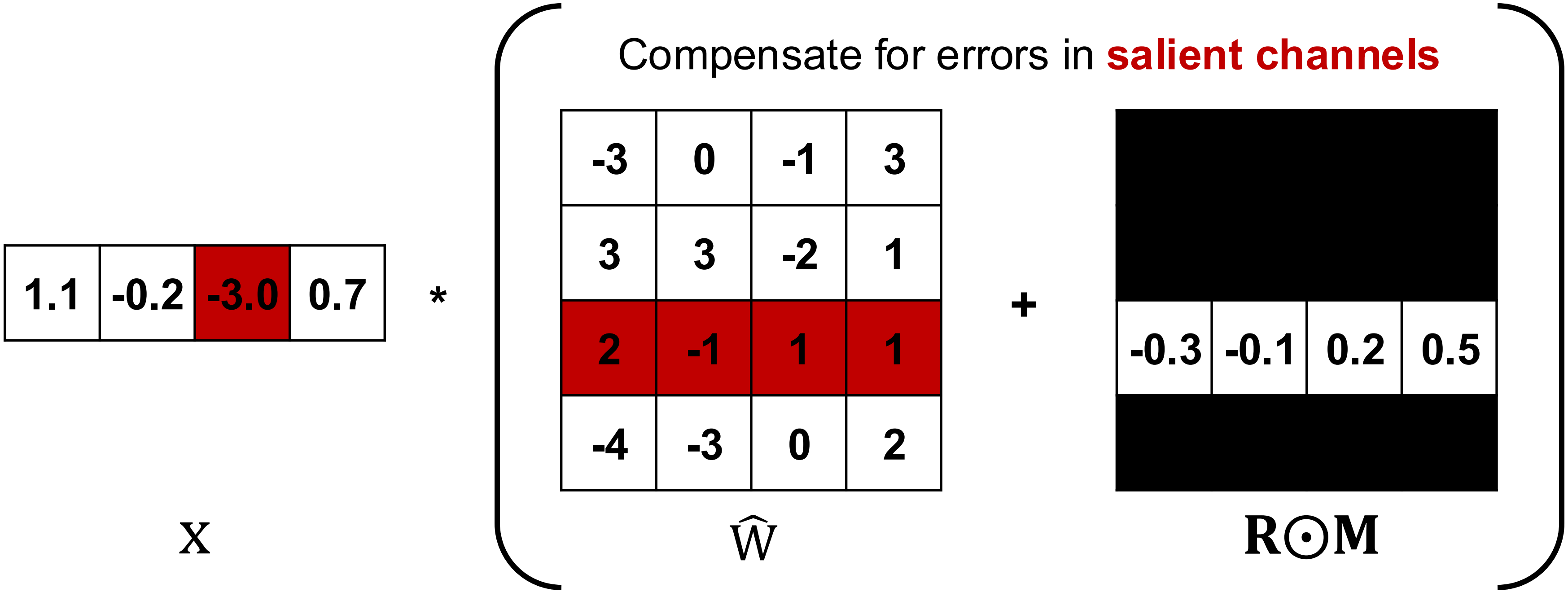
Opportunity: Not All Channels are Equally Important

$$\underbrace{\begin{bmatrix} 1.1 & -0.2 & -3.0 & 0.7 \end{bmatrix}}_{\mathrm{X}} * \underbrace{\begin{bmatrix} -3 & 0 & -1 & 3 \\ 3 & 3 & -2 & 1 \\ 2 & -1 & 1 & 1 \\ -4 & -3 & 0 & 2 \end{bmatrix}}_{\widehat{\mathrm{W}}} \leftarrow \underbrace{\begin{bmatrix} -3.2 & 0.2 & -1.0 & 3.1 \\ 2.8 & 3.7 & -2.0 & 1.3 \\ 1.7 & -1.1 & 1.2 & 0.5 \\ -4.0 & -2.7 & 0.4 & 2.1 \end{bmatrix}}_{\mathrm{W}}$$

Opportunity: Not All Channels are Equally Important



Opportunity: Not All Channels are Equally Important



Challenge: Dynamic Nature of Activation Outliers

Decoding Step i

$$\underbrace{\begin{bmatrix} 1.1 & -0.2 & -3.0 & 0.7 \end{bmatrix}}_{\mathrm{X}} \;*\; \underbrace{\begin{bmatrix} -3 & 0 & -1 & 3 \\ 3 & 3 & -2 & 1 \\ 2 & -1 & 1 & 1 \\ -4 & -3 & 0 & 2 \end{bmatrix}}_{\widehat{\mathrm{W}}}$$

Challenge: Dynamic Nature of Activation Outliers

Decoding Step $i + 1$

4.1	-1.2	0.8	0.3

X

*

-3	0	-1	3
3	3	-2	1
2	-1	1	1
-4	-3	0	2

$\widehat{W}$

Challenge: Dynamic Nature of Activation Outliers

Decoding Step $i + 2$

-0.1	0.8	1.1	-3.7

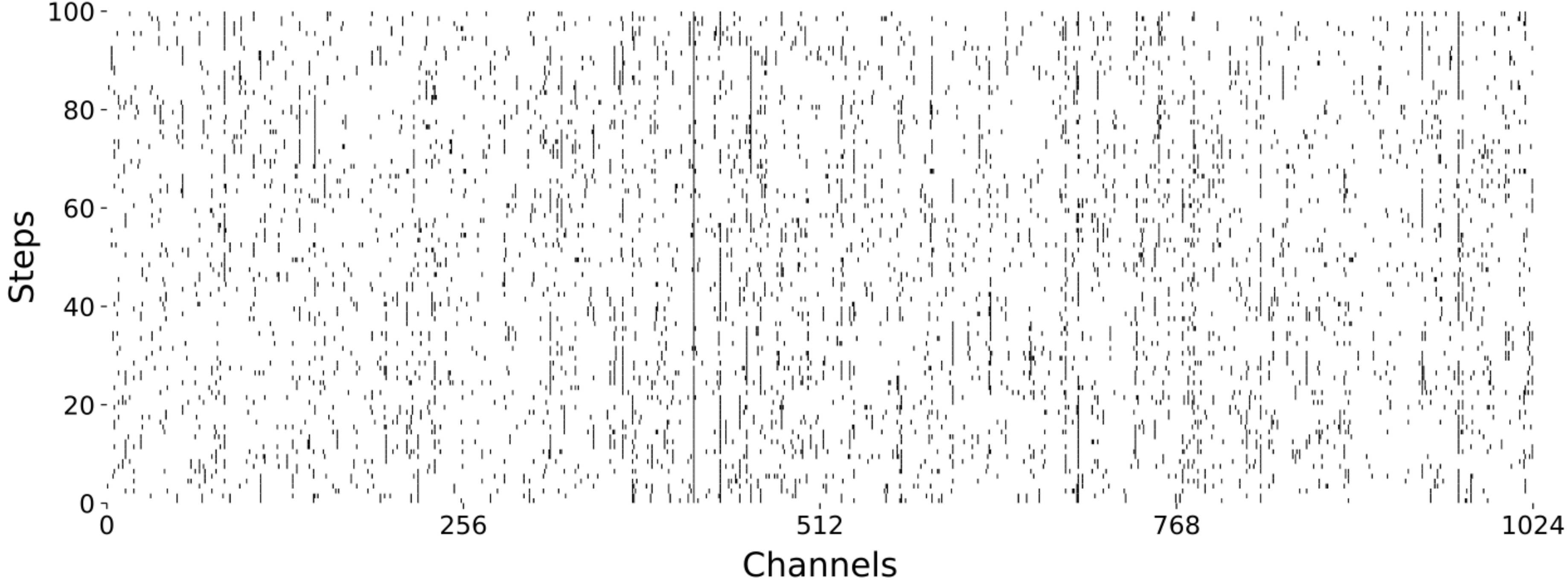
X

*

-3	0	-1	3
3	3	-2	1
2	-1	1	1
-4	-3	0	2

$\hat{W}$

Challenge: Dynamic Nature of Activation Outliers

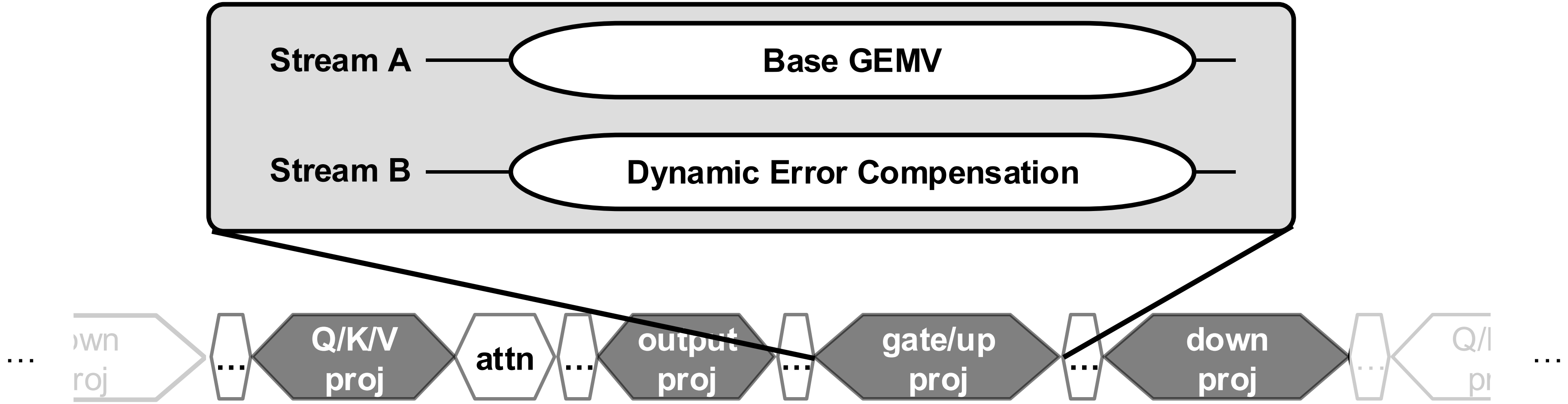


Distribution of activation outliers (<u>top 5%</u>) across 100 decoding steps

Department of CSE
Seoul National University

DecDEC: Decoding with Dynamic Error Compensation

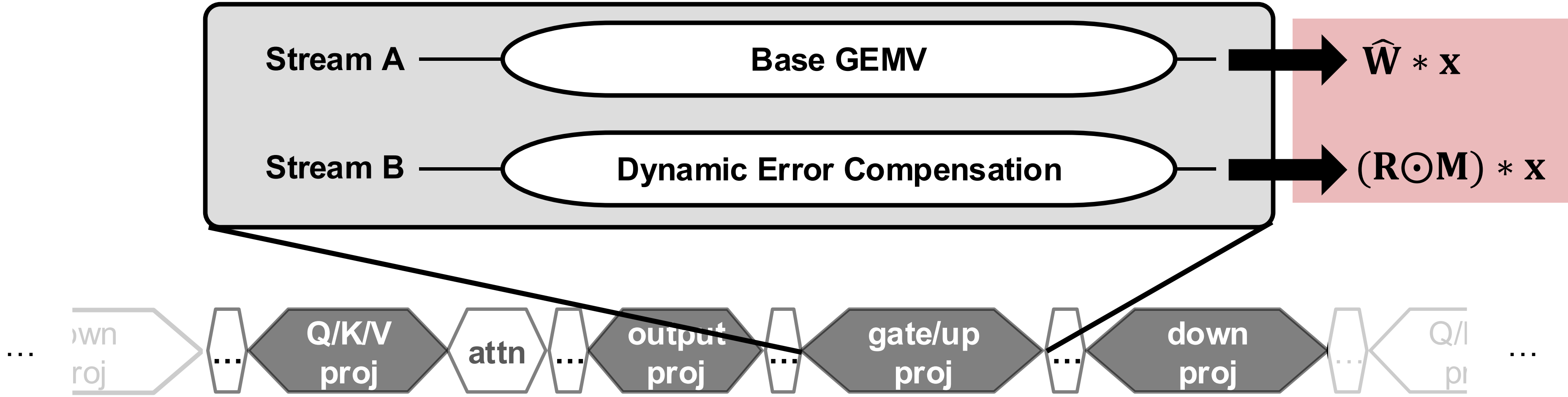
Inference system for quantized LLMs that performs **Dec**oding with **D**ynamic **E**rror **C**ompenstation



Augment each linear layer with **dynamic error compensataion (DEC)**

DecDEC: Decoding with Dynamic Error Compensation

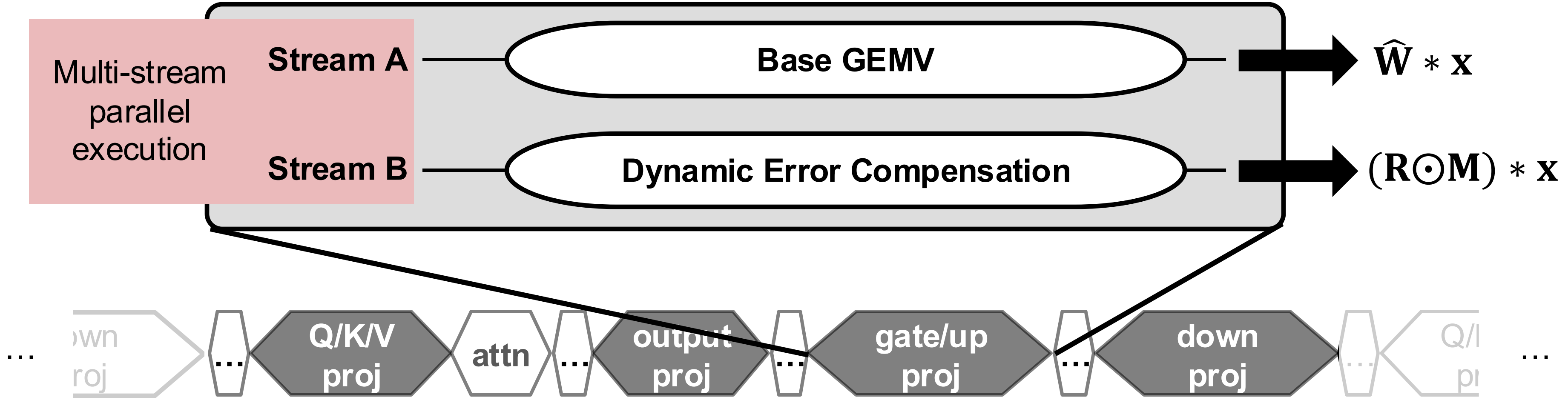
Inference system for quantized LLMs that performs **Dec**oding with **D**ynamic **E**rror **C**ompenstation



Augment each linear layer with **dynamic error compensataion (DEC)**

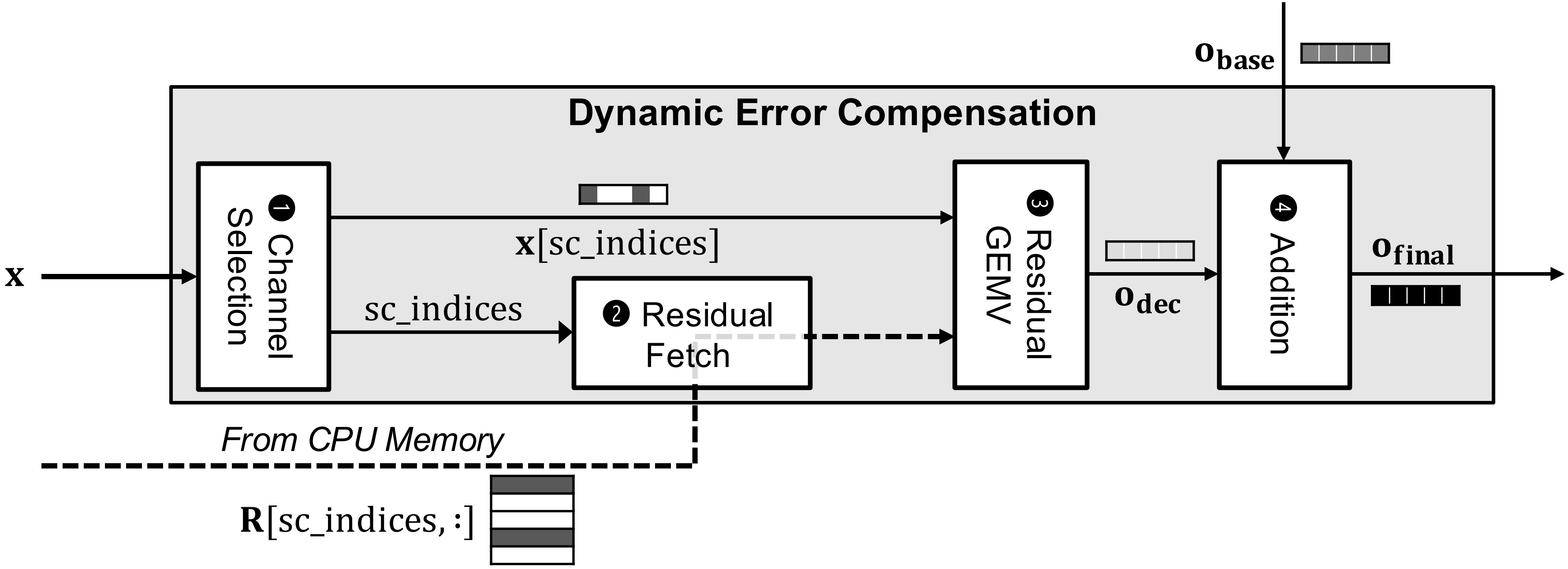
DecDEC: Decoding with Dynamic Error Compensation

Inference system for quantized LLMs that performs **Dec**oding with **D**ynamic **E**rror **C**ompenstation

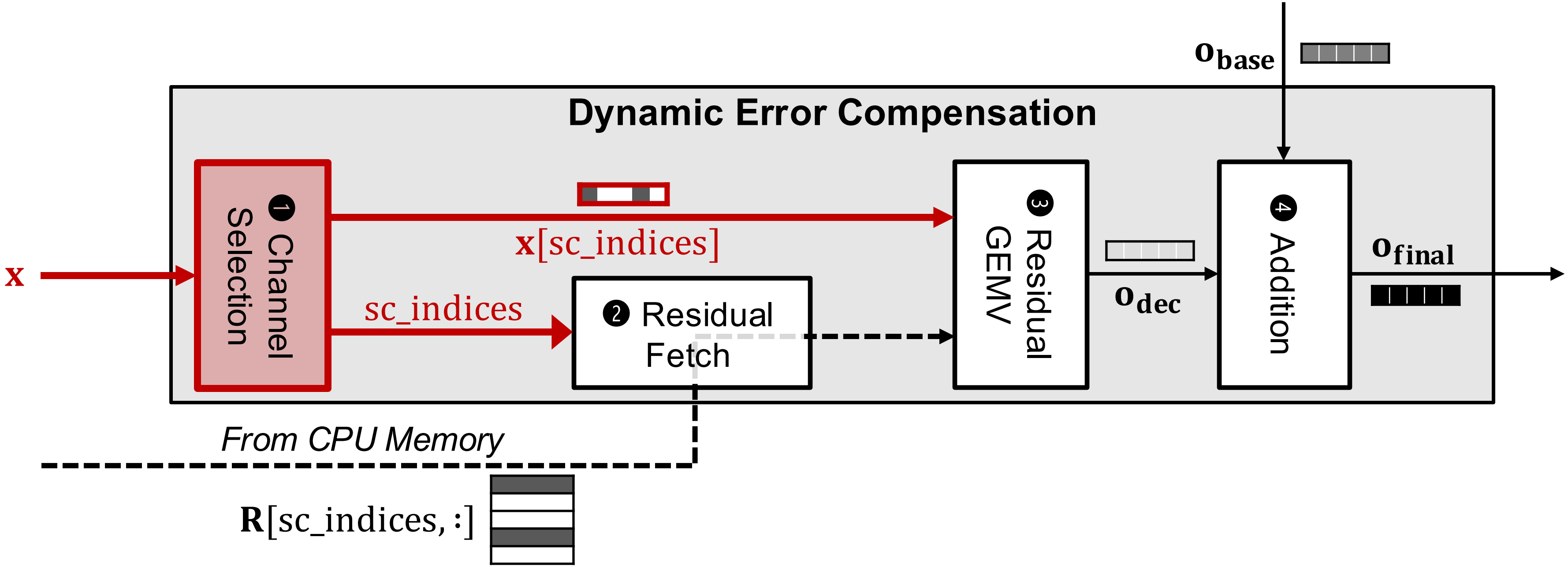


Augment each linear layer with **dynamic error compensataion (DEC)**

Four Steps of Dynamic Error Compensation

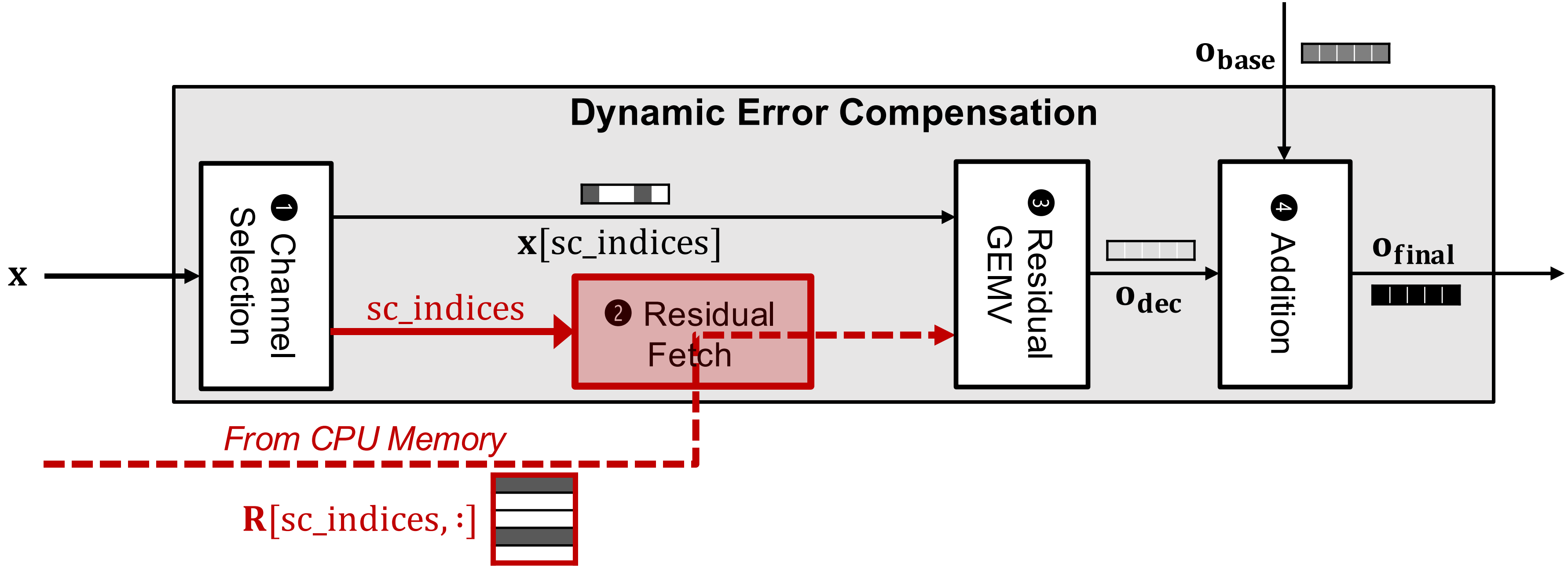


Four Steps of Dynamic Error Compensation



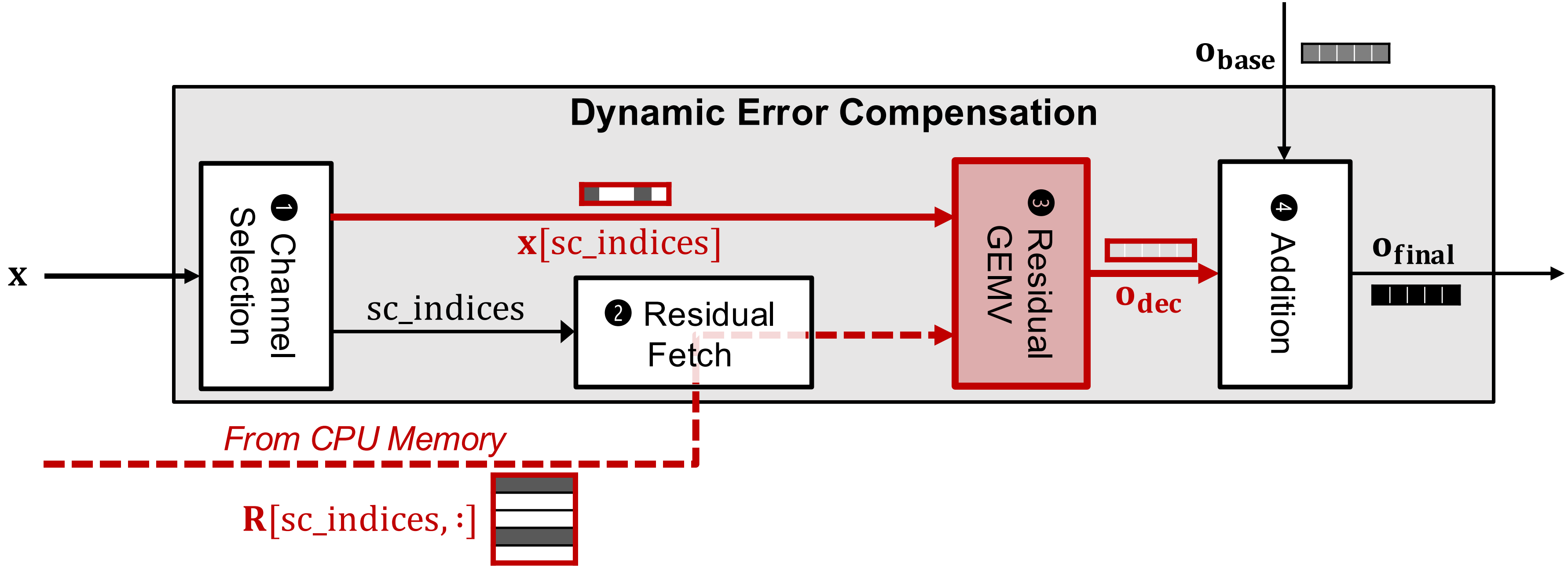
① Perform Top-K on the input activation vector ($\mathbf{x}$)

Four Steps of Dynamic Error Compensation



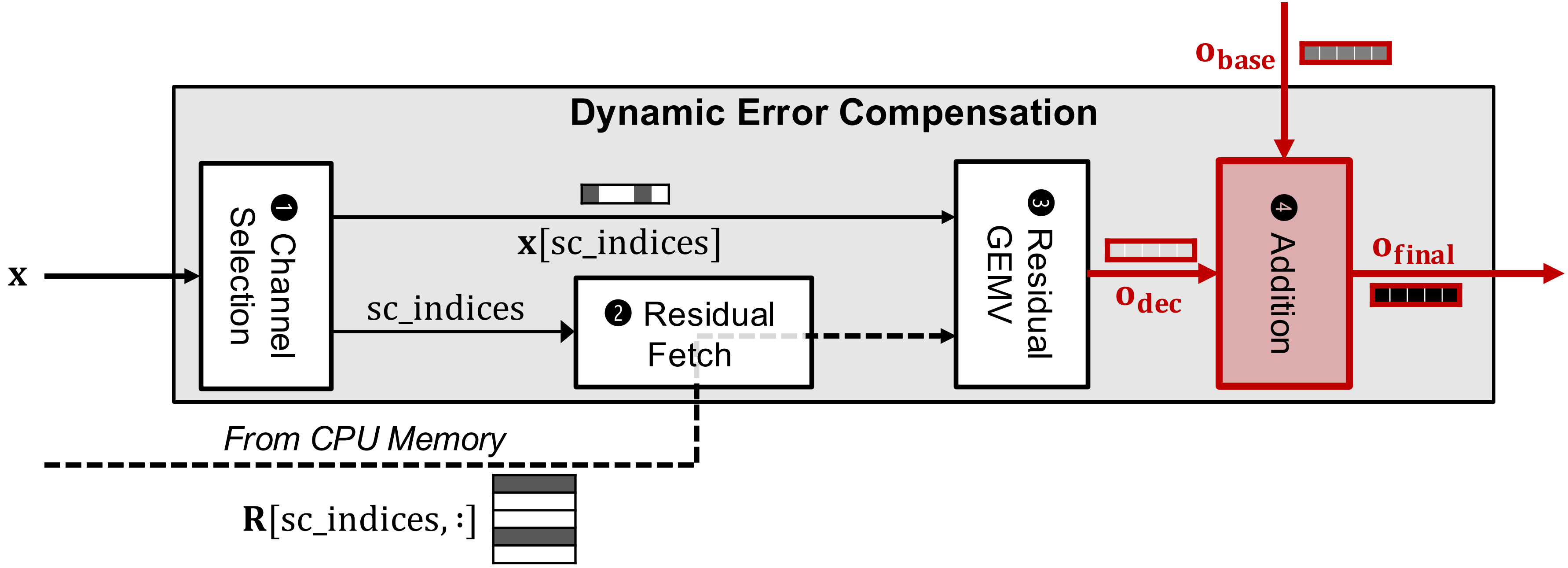
② Fetch residuals for the selected channels from CPU ($\mathbf{R}$[sc_indices, :])

Four Steps of Dynamic Error Compensation



③ Multiply the input activation by the selected residuals

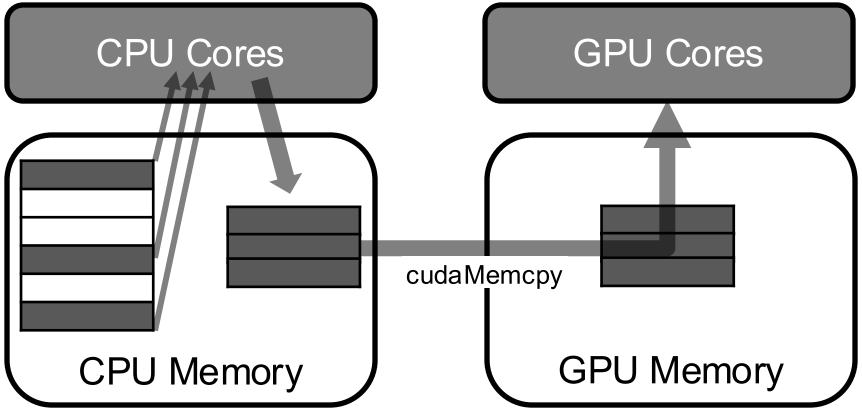
Four Steps of Dynamic Error Compensation



④ Add base GEMV result ($\mathbf{o_{base}}$) & residual GEMV result ($\mathbf{o_{dec}}$)

Key Implementation Point: Zero-Copy Residual Fetch

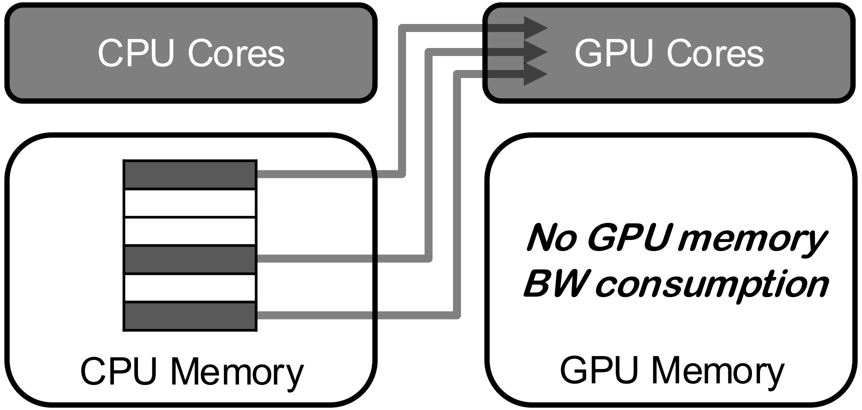
DMA-based Approach



👍 High BW util for bulk transfer

👎 Long latency

Zero-Copy-based Approach



👍 Low latency
👍 Fine-grained, cacheline-sized data access

👎 GPU core consumption

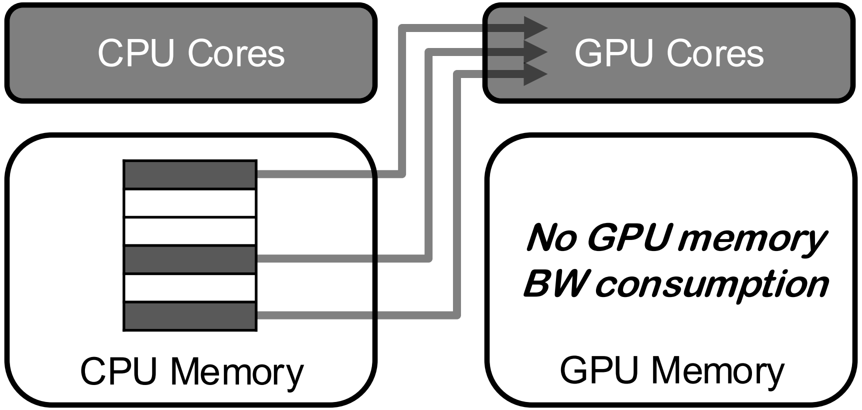
Key Implementation Point: Zero-Copy Residual Fetch

Zero-Copy is the answer!

* Need low-latency, fine-grained data access

* GPU core consumption is OK,
while BW contention is undesirable
(Base GEMV is memory-bound)

Zero-Copy-based Approach



👍 Low latency

👍 Fine-grained, cacheline-sized data access

👎 GPU core consumption

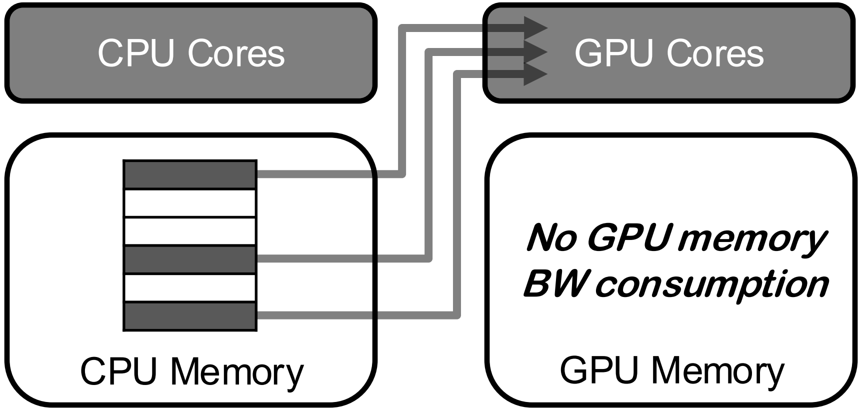
Key Implementation Point: Zero-Copy Residual Fetch

Zero-Copy is the answer!

* Need low-latency, fine-grained data access

* GPU core consumption is OK,
while BW contention is undesirable
(Base GEMV is memory-bound)

Zero-Copy-based Approach



- 👍 Low latency
- 👍 Fine-grained, cacheline-sized data access
- 👎 GPU core consumption

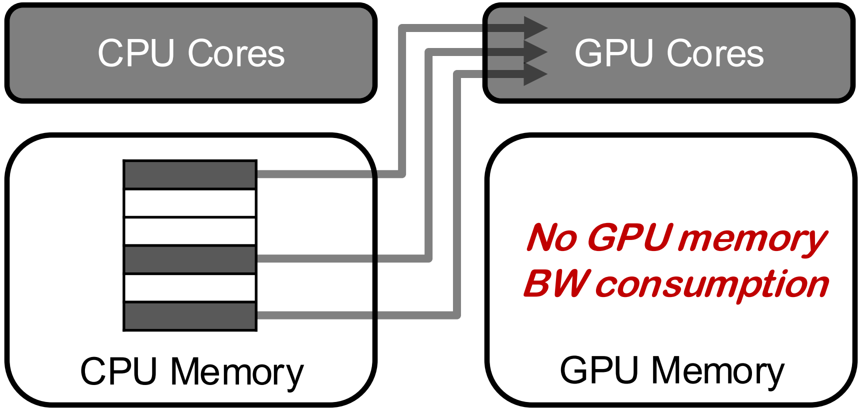
Key Implementation Point: Zero-Copy Residual Fetch

Zero-Copy is the answer!

* Need low-latency, fine-grained data access

* GPU core consumption is OK,
while BW contention is undesirable
(Base GEMV is memory-bound)

Zero-Copy-based Approach



👍 Low latency

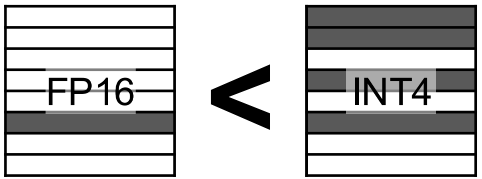
👍 Fine-grained, cacheline-sized data access

👎 GPU core consumption

Department of CSE
Seoul National University

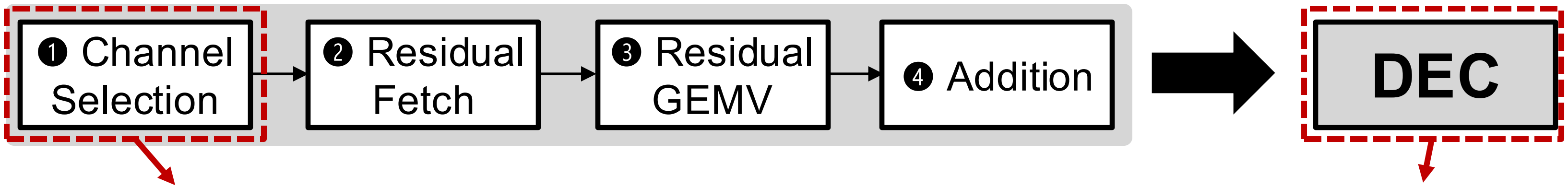
Other Optimizations

- 4-bit residual quantization



Introduces approximation errors,
but enables fetching more channels

- GPU kernel optimizations

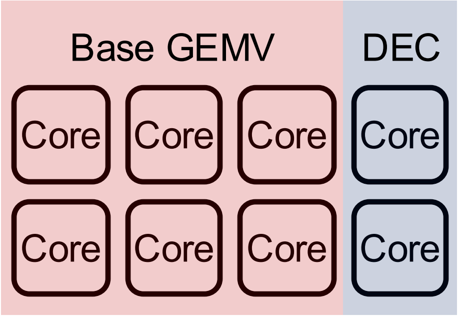


GPU-friendly approximiate
Top-K selection

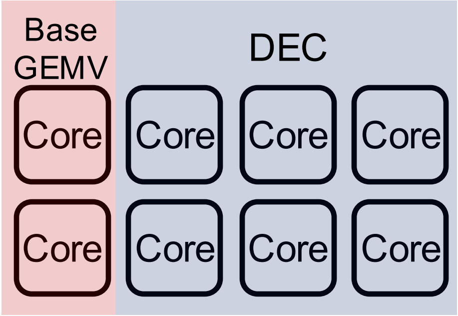
Full kernel fusion

DecDEC Parameter Tuner

- Two system parameters should be carefully tuned
  - ① $\boldsymbol{n_{tb}}$: # of thread blocks (≈ # of GPU cores) to allocate to DEC



Too small n_{tb} (i.e., 2) underutilizes PCIe BW



Too large n_{tb} (i.e., 6) slows down the base GEMV

  - ② $\boldsymbol{k}$: # channels to fetch
    → Larger is better, up to the point it incurs too high latency overhead

DecDEC Parameter Tuner

Model

Device

Latency budget
(slowdown rate relative to baseline w/o DecDEC)

→

DecDEC Parameter Tuner

(heuristic search under latency budget)

→

n_{tb} and k
for each layer

Evaluation

Evaluation

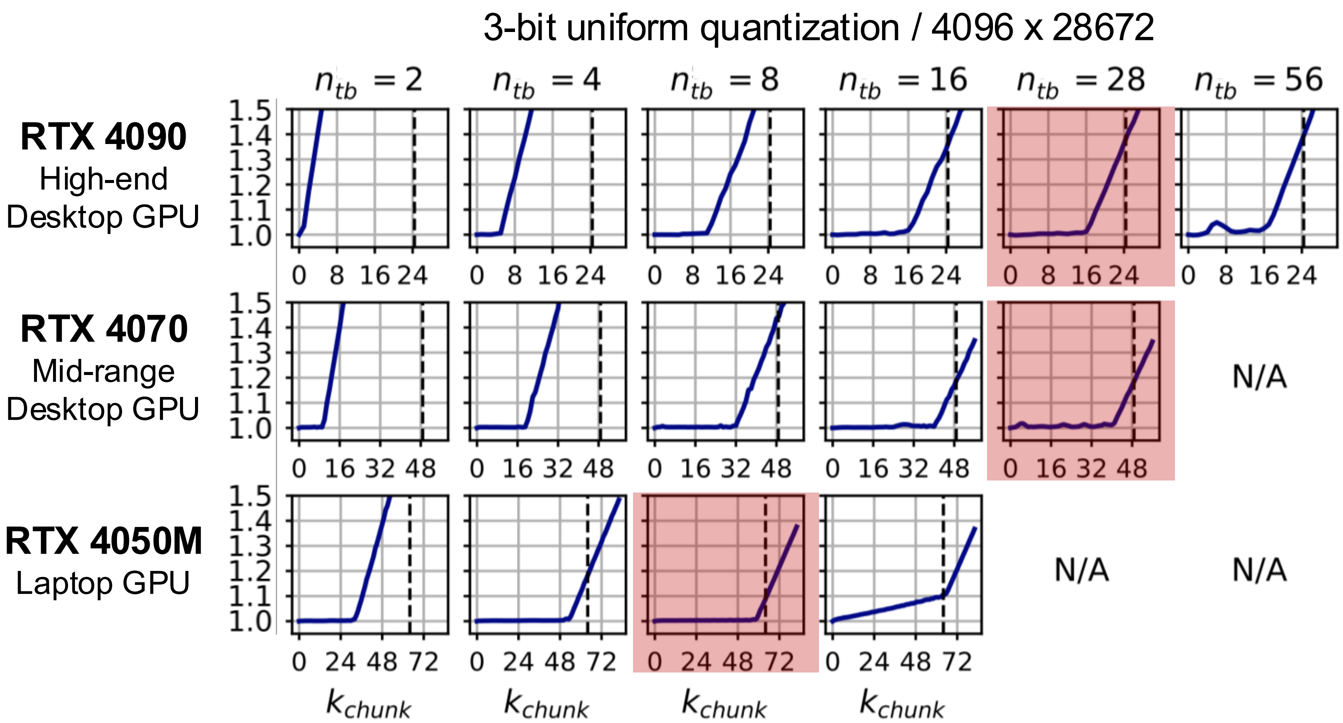
- **Key Result #1:**
  DEC is effectively overlapped with base GEMV using appropriate n_{tb} and k

- **Key Result #2:**
  DecDEC significantly improves quality with limited latency overhead

Evaluation

- **Key Result #1:**
  DEC is effectively overlapped with base GEMV using appropriate n_{tb} and k
- **Key Result #2:**
  DecDEC significantly improves quality with limited latency overhead

DecDEC Kernel Evaluation



With appropriate n_{tb}, DecDEC incurs almost no overhead up to a certain value of k (k_{chunk}: k per 1024 channels)

Department of CSE
Seoul National University

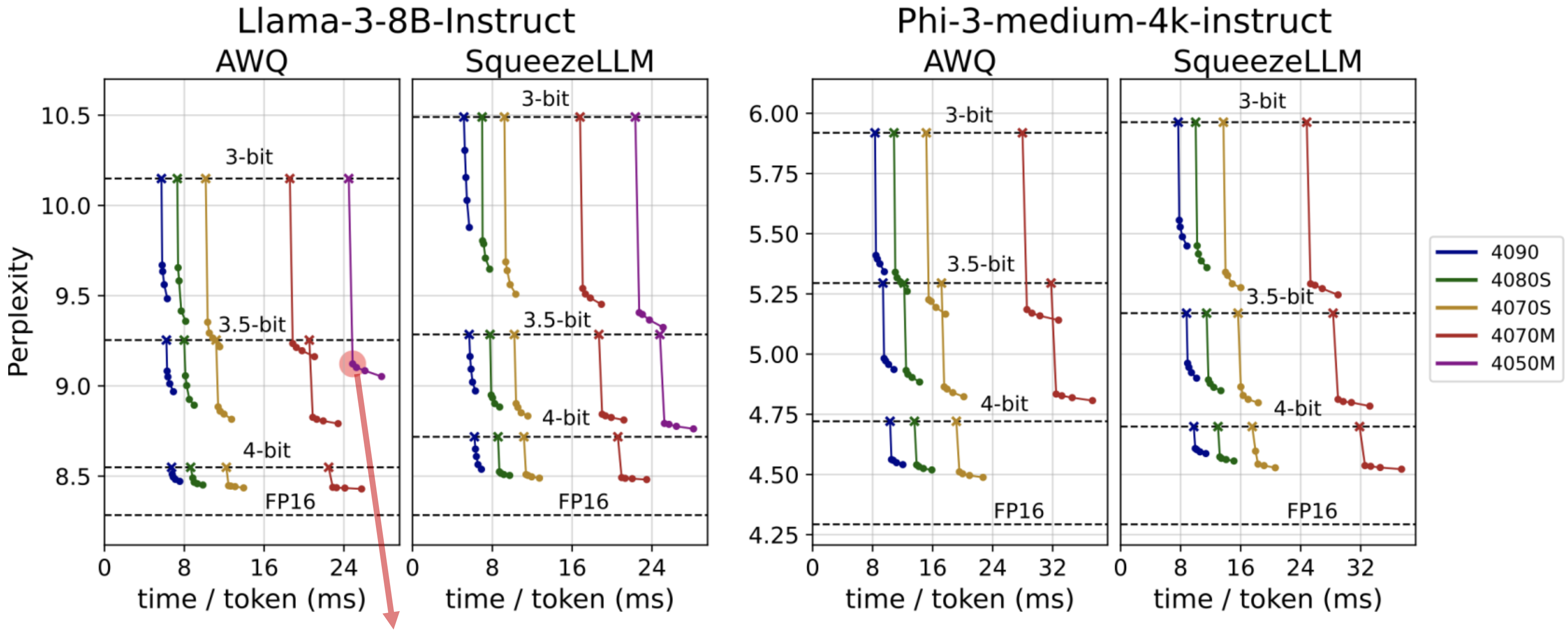
Evaluation

- **Key Result #1:**
  DEC is effectively overlapped with base GEMV using appropriate n_{tb} and k

- **Key Result #2:**
  DecDEC significantly improves quality with limited latency overhead

End-to-End Evaluation

Perplexity vs. Latency under 2.5%, 5%, 10%, and 20% target slowdown rate



GPU Mem.	Latency	Quality
3-bit	~3-bit (1.7% ↑)	Better than 3.5-bit