BLADE: Boosting LLM Decoding's Communication Efficiency in DRAM-based PIM

Yilong Zhao (Speaker), Fangxin Liu, Zongwu Wang, Mingjian Li, Mingxing Zhang, Chixiao Chen and Li Jiang

Shanghai Jiao Tong University, Shanghai Qi Zhi Institute,

Tsinghua University, Fudan University

ASP-DAC 2026

Outline

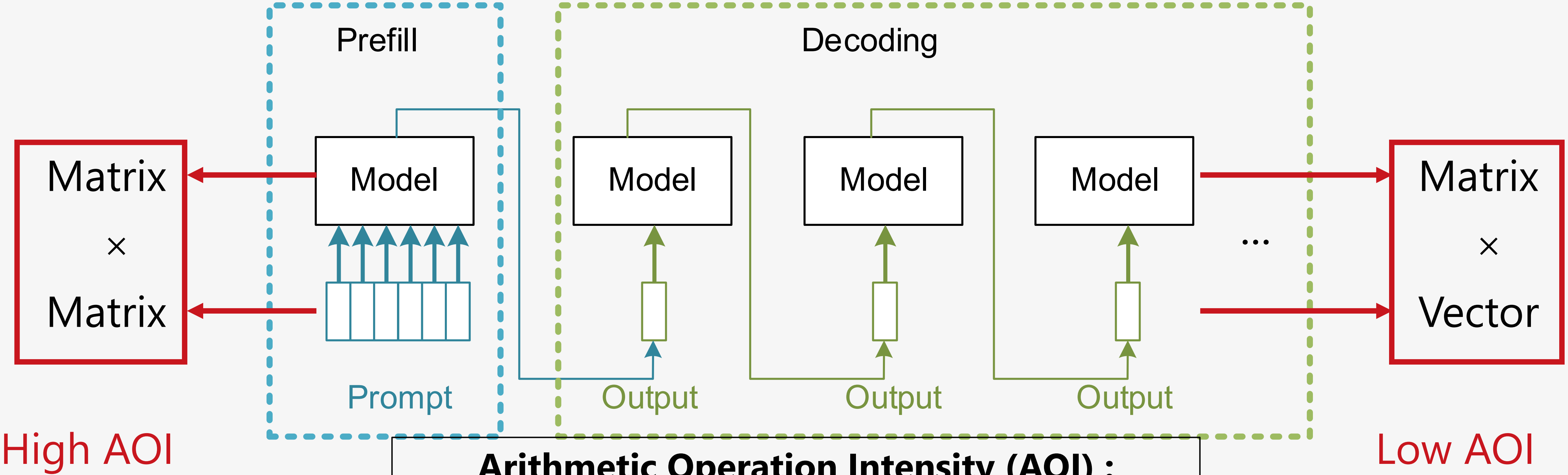
- **Context:** Large Language Model (LLM) inference, **Prefill-Decoding Disaggregation**
  - Prefill->GPU; Decoding -> DRAM-based **Processing-in-Memory (PIM)**
- **Challenges on Communication & Solutions:**
  - Challenge 1: High KV Cache Transpose Latency During “GPU->PIM” KV Cache Transfer
  - Solution 1: **Transpose on Transfer Method**
  - Challenge 2: Static PIM Parallelism Fails to Balance Communication & Computation
  - Solution 2: **Dynamical Parallelism Scaling Method**
- **Results:**
  - Transpose on Transfer 1.54x; Dynamical Parallelism Scaling 2.09x

Background:

Large Language Models
PIM-based P-D Disaggregation

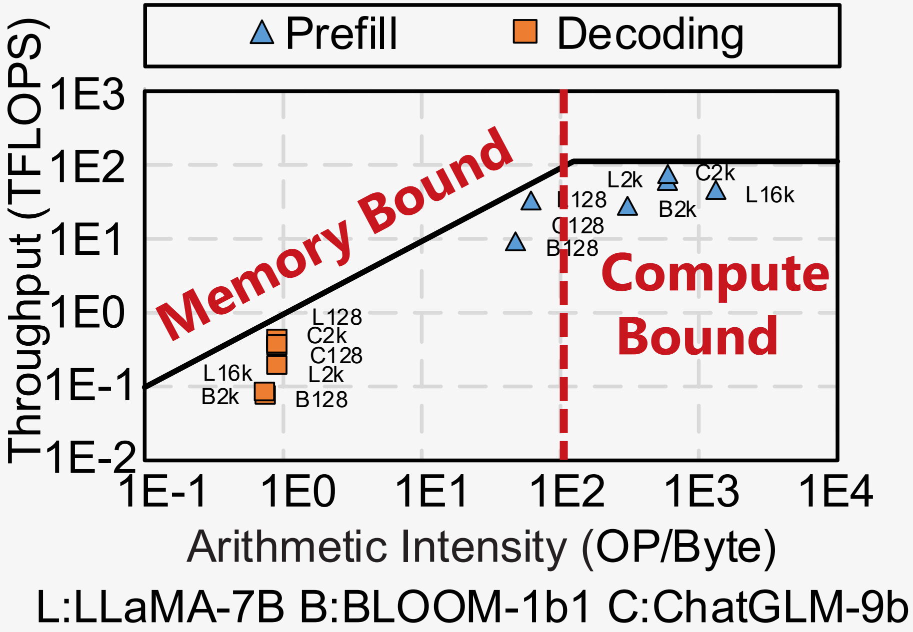
Prefill-Decoding Disaggregation

- Two phases of Large Language Model (LLM) inference: **Prefill** & **Decoding**



Arithmetic Operation Intensity (AOI) :
#operation / #memory access count (OP/Byte)

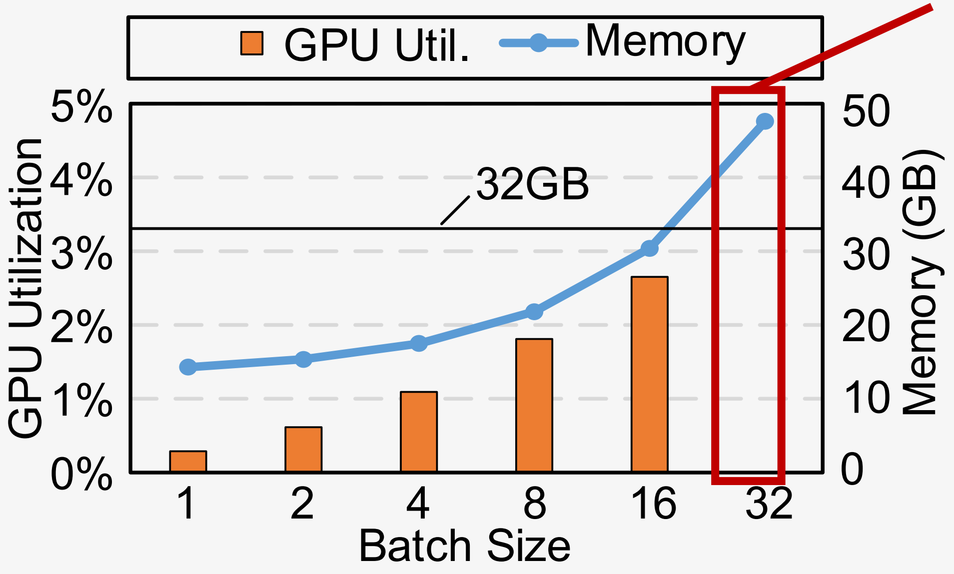
AOI of Prefill / Decoding



(a)

Prefill & Decoding on V100 GPU

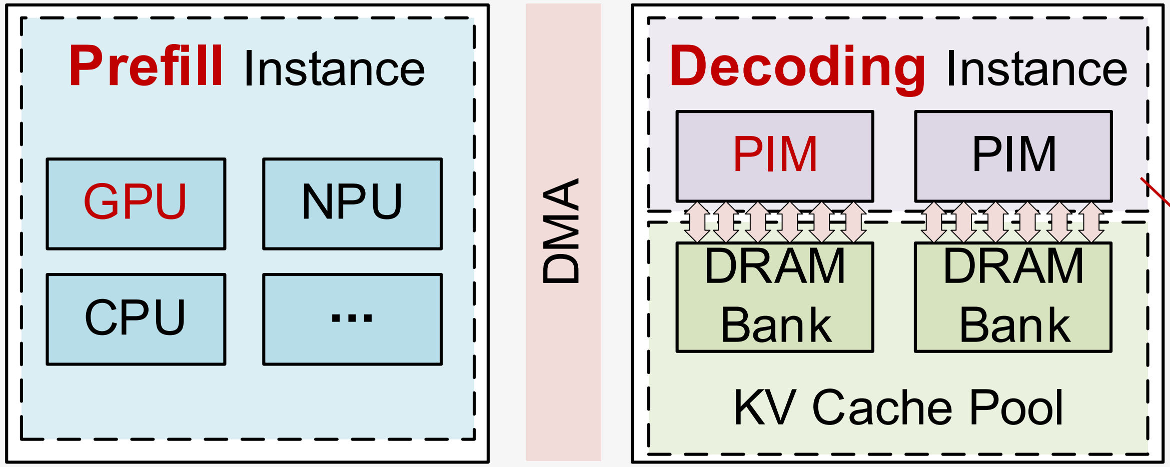
Memory Footprint Exceeds 32GB!
GPU Util still < 3%



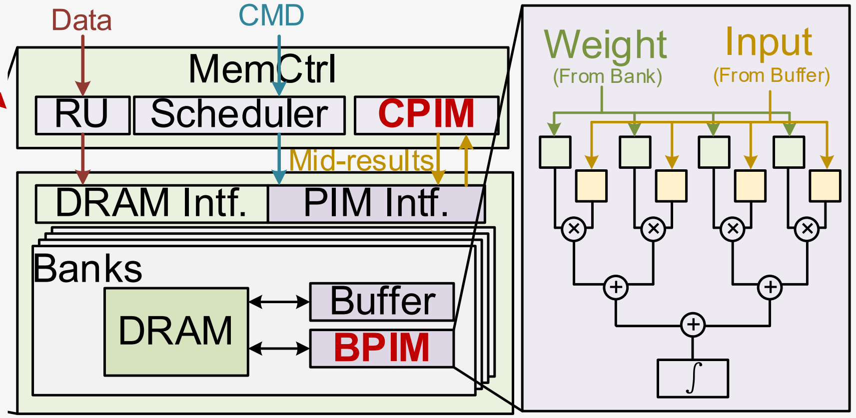
(b)

Batching does not work for Decoding

PIM-based PD Disaggregation



Processing-in-Memory (PIM):
Integrate computational unit inside DRAM memory



Prefill: High AOI

GPU: High Computing Power

Decoding: Low AOI

PIM: High Memory Bandwidth

Two-level PIM:
BPIM: In Bank for MAC
CPIM: In MemCtrl for mid-result reducing

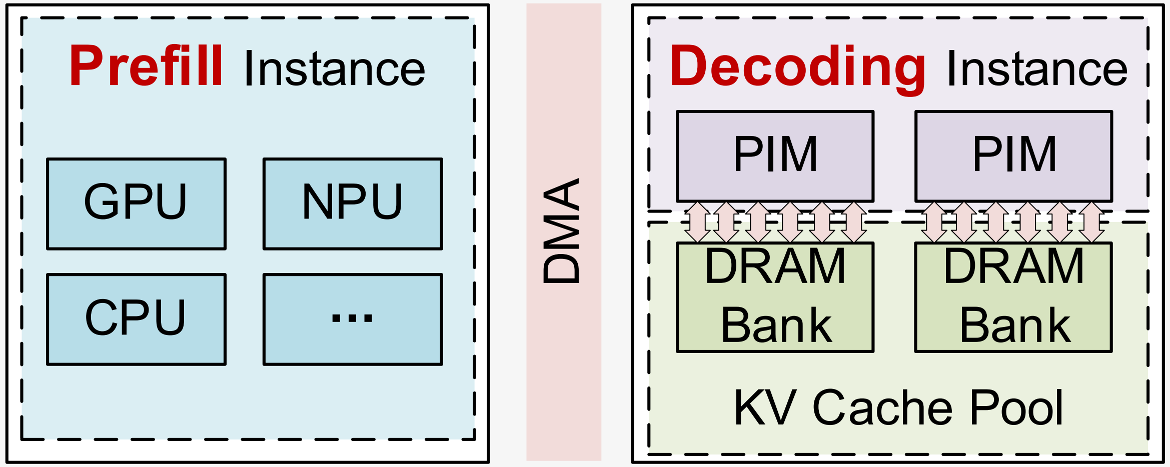
Challenges & Solutions

2 Challenges on Communication

Our Solution – BLADE Architecture

Communication Challenge 1 – V-Cache Transpose (1)

- During Key-Value Cache (KV Cache) Transfer



Step 1: In Prefill Instance
Generate **Output** & **KV Cache**

Step 2: Transfer **KV Cache** to PIM side

Step 3: In Decoding Instance
Consume **KV Cache** and generate output

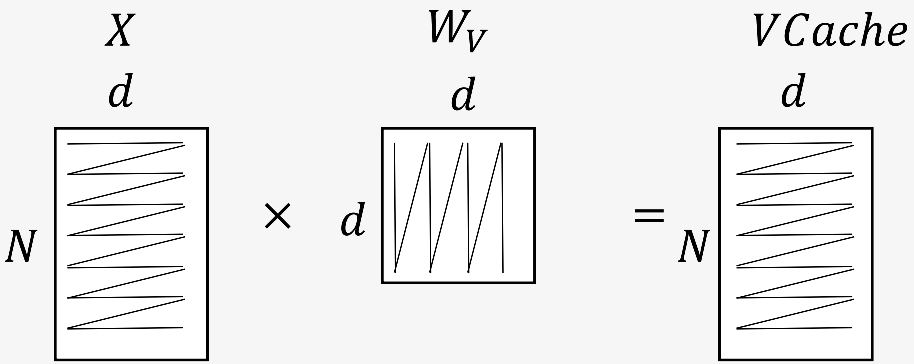
Communication Challenge 1 – V-Cache Transpose (2)

(Prefill phase) Producing Value Cache:

- $VCache = X \times W_V$

$(N \times d)$ $(N \times d)$ $(d \times d)$

Token Num — Head Size

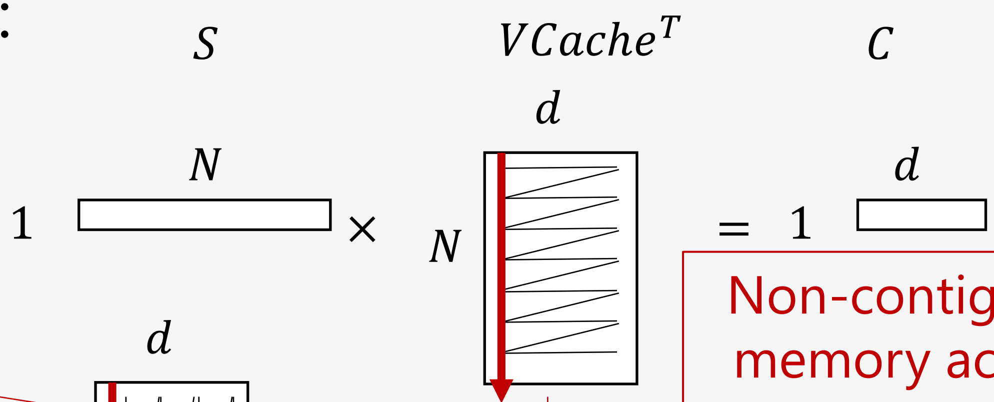


Contiguous along d dimension

(Decoding phase) Consuming Value Cache:

- $C = S \times VCache^T$

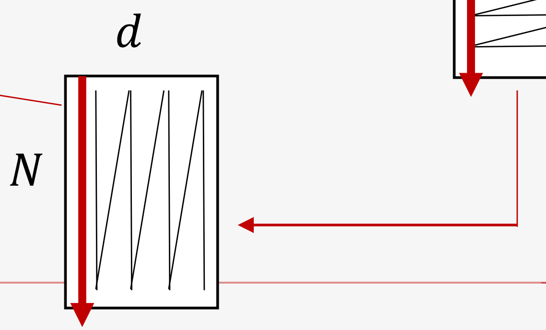
$(1 \times d)$ $(1 \times N)$ $(N \times d)$



Non-contiguous memory access: Low DRAM bandwidth

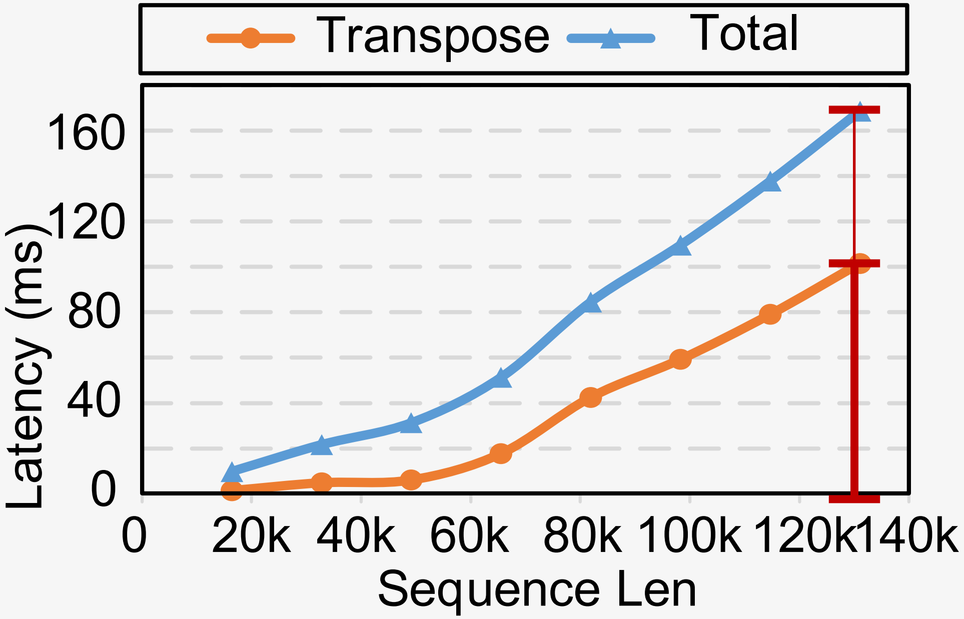
Preferred layout

(Require GPU Transpose!)



Communication Challenge 1 – V-Cache Transpose (3)

- Inefficiency of GPU Transpose:

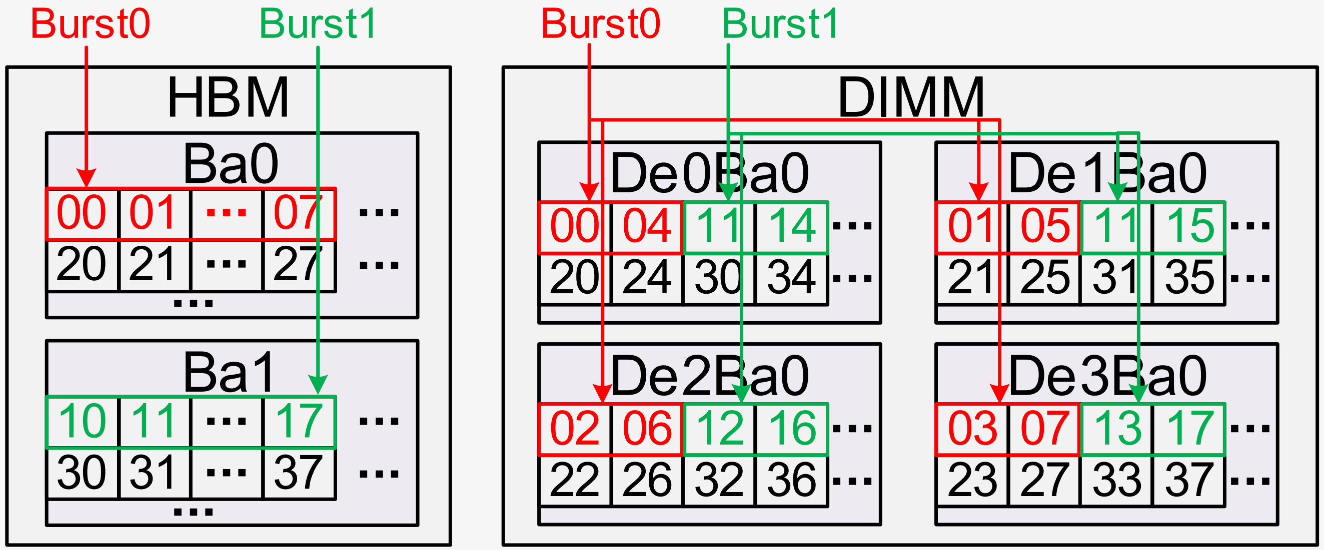


Over 78% of KV Cache transfer time

Solution to Challenge 1: Transpose-on-Transfer (1)

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

- Utilize the different access granularity between CPUs and PIM units:

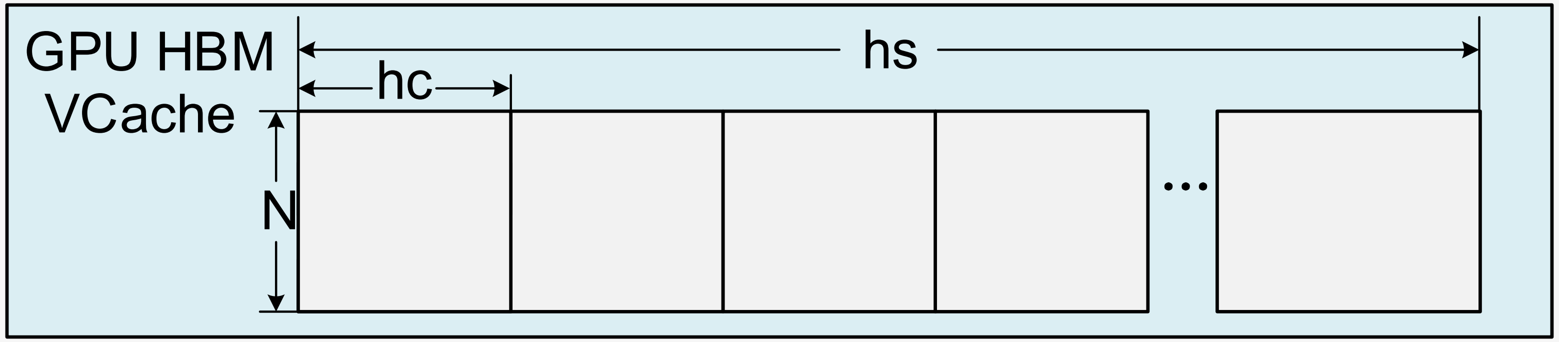


HBM Burst:
Contiguous inside one Bank

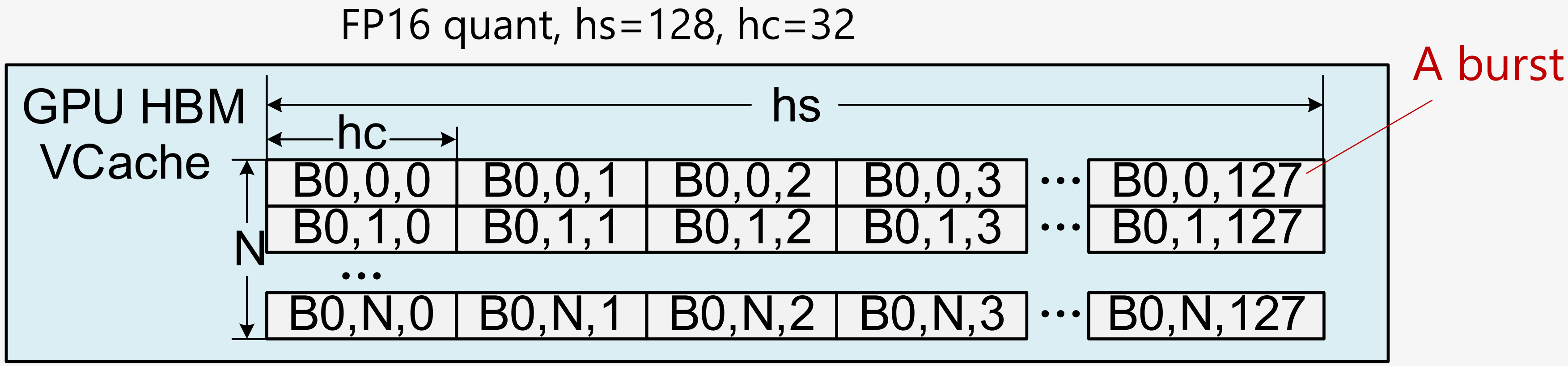
DIMM Burst:
Interleaved between Devices
(Device also called "Chip")

Solution for Challenge 1: Transpose-on-Transfer (2)

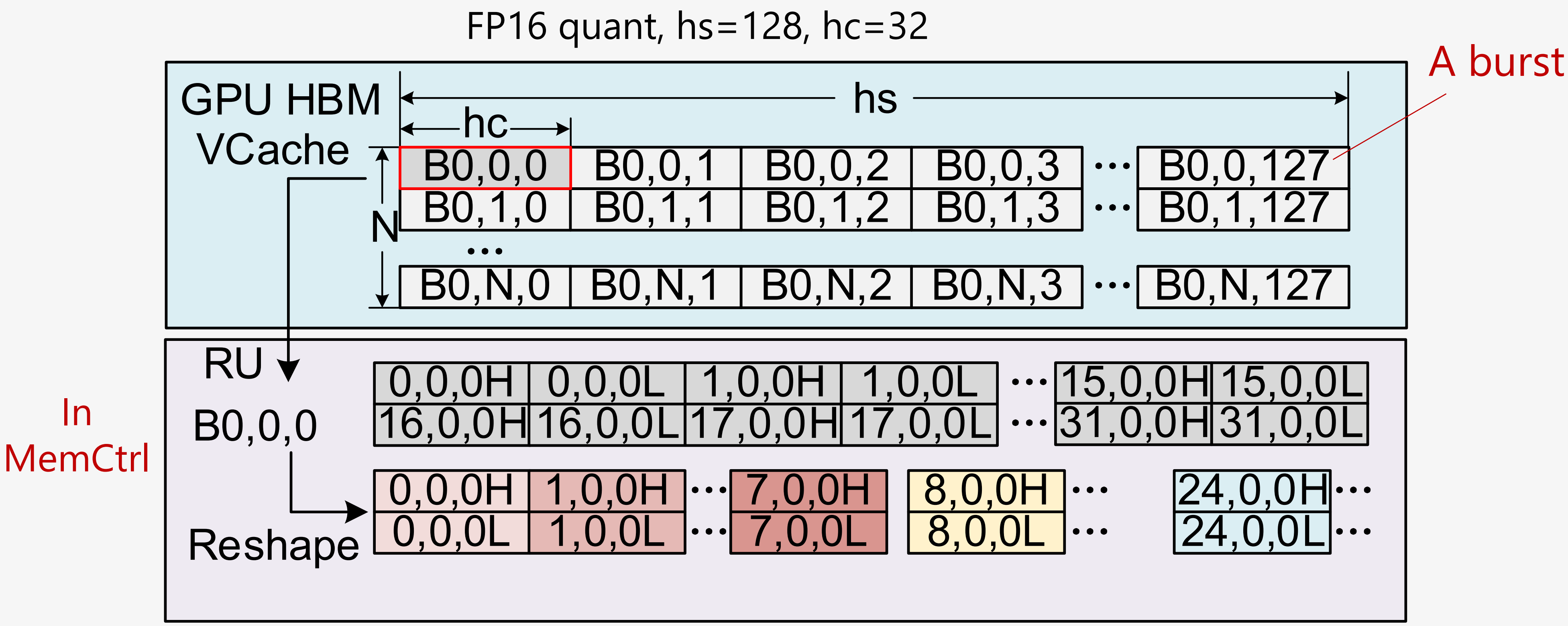
FP16 quant, hs=128, hc=32



Solution for Challenge 1: Transpose-on-Transfer (2)

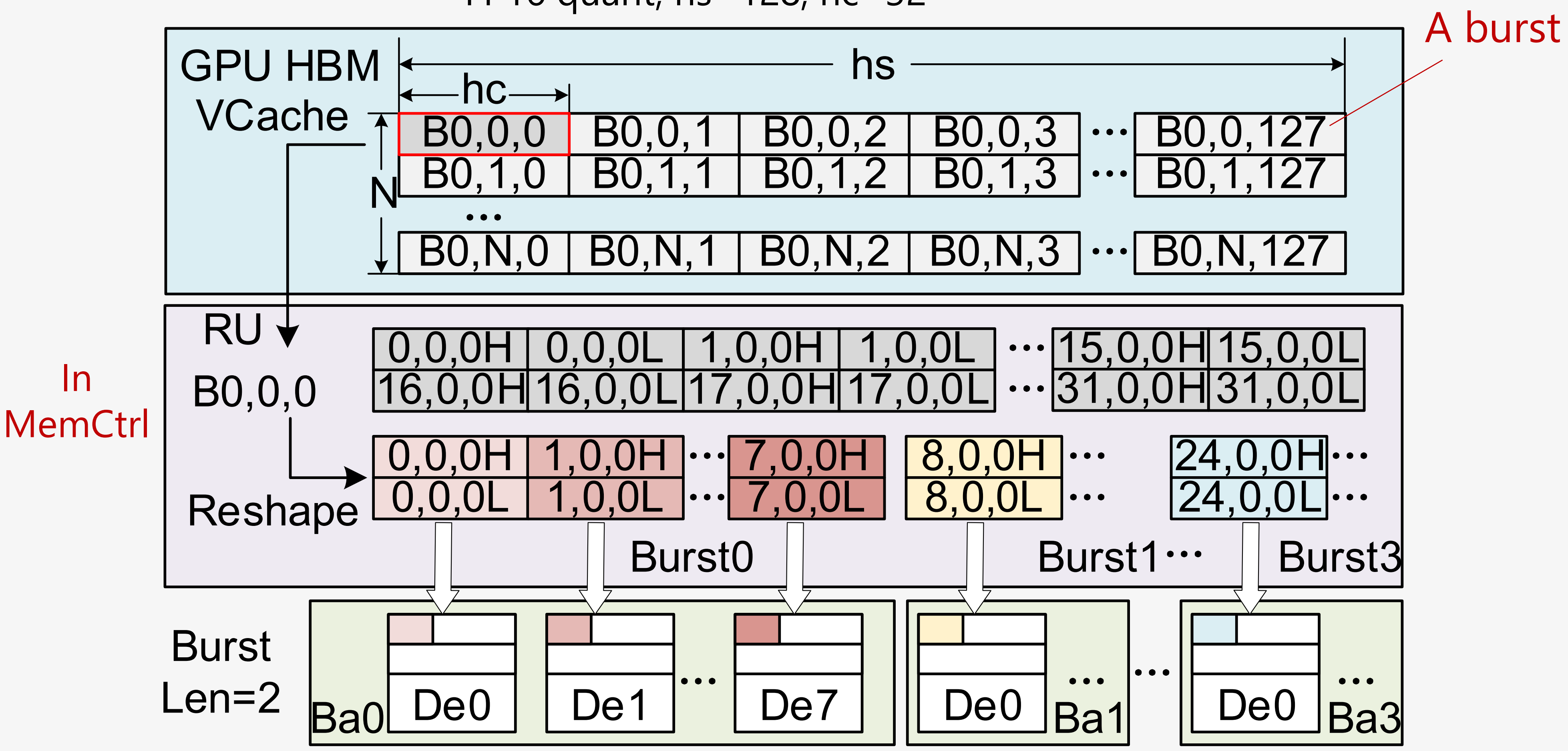


Solution for Challenge 1: Transpose-on-Transfer (2)

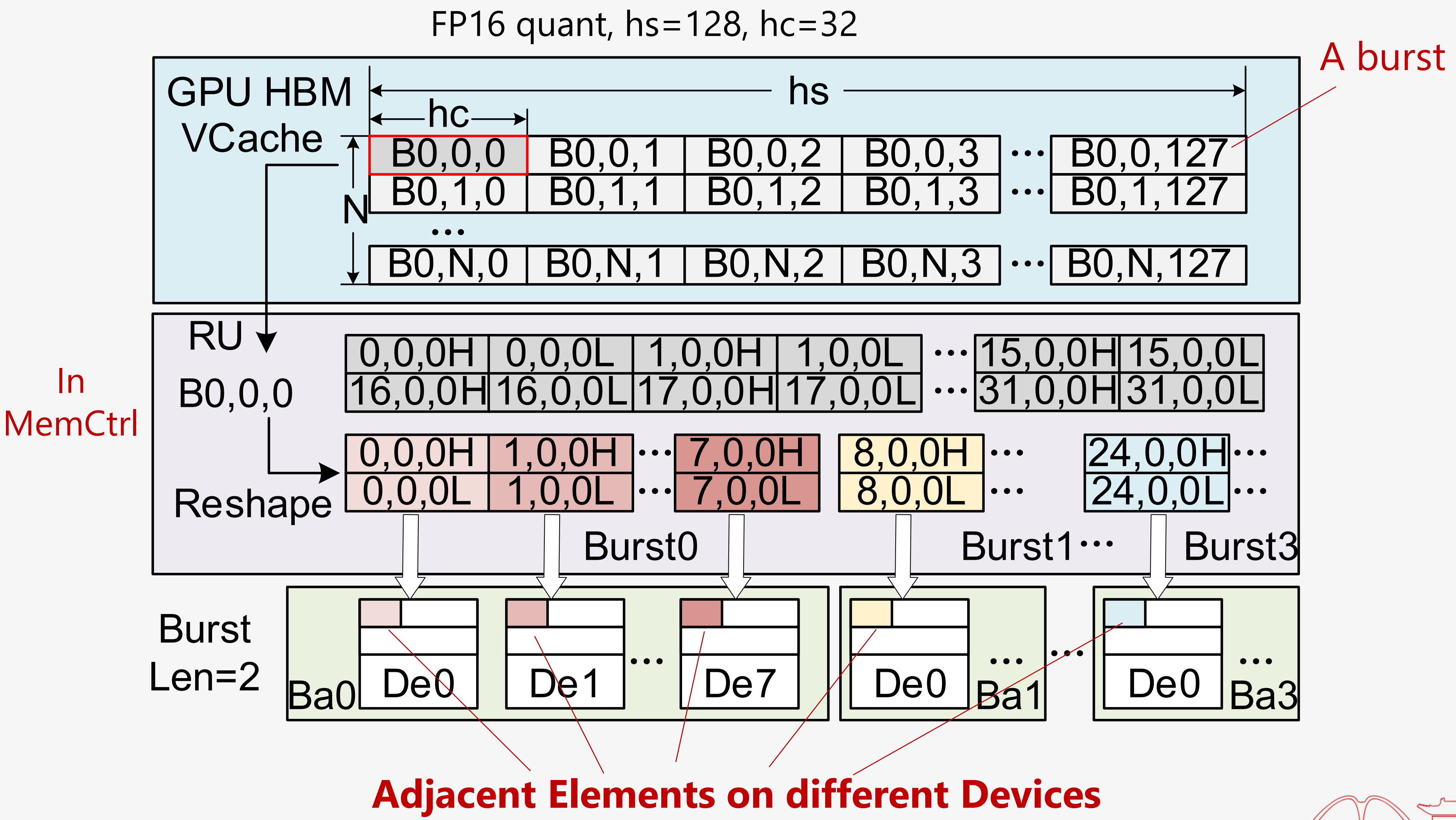


Solution for Challenge 1: Transpose-on-Transfer (2)

FP16 quant, hs=128, hc=32

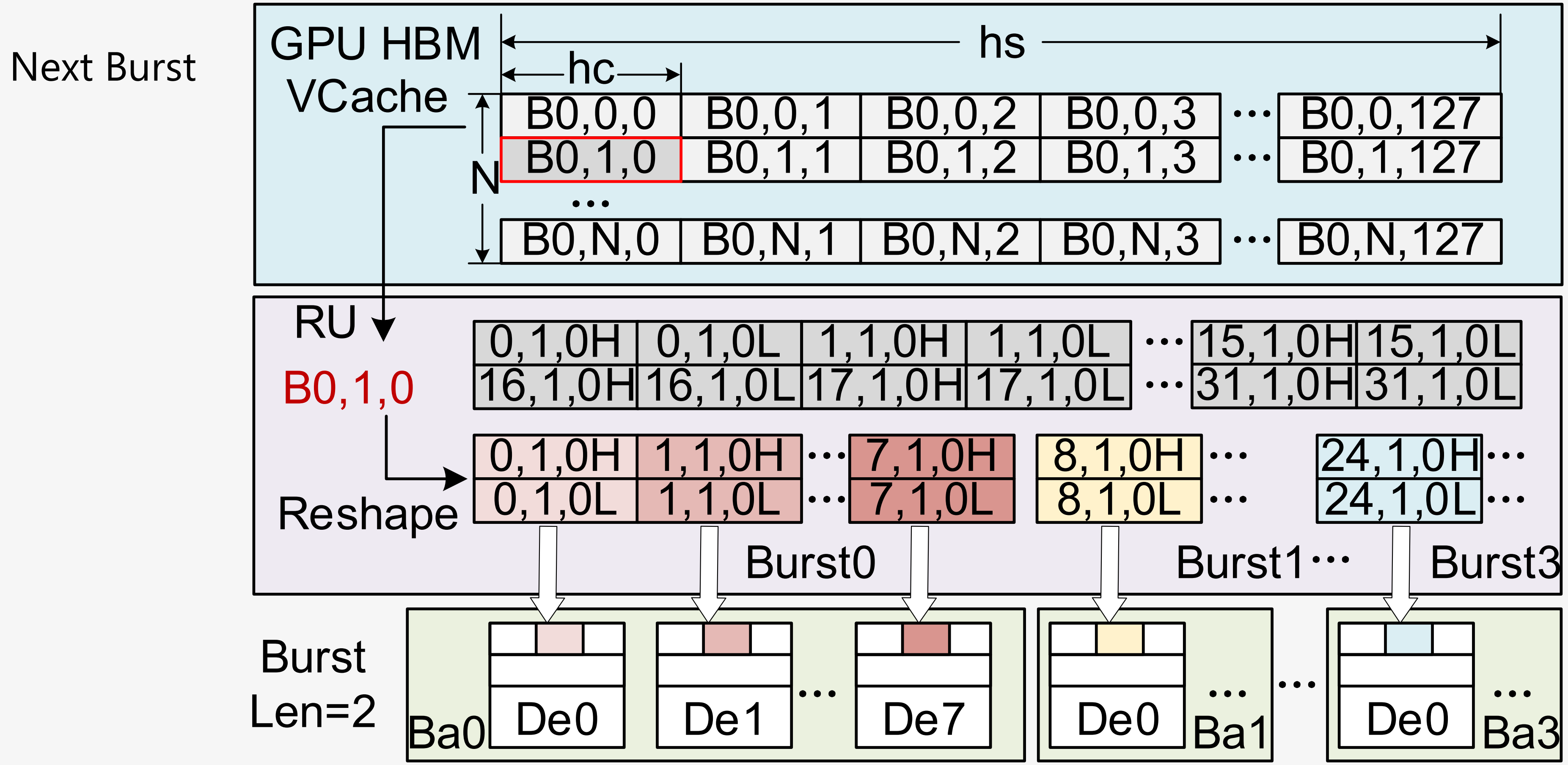


Solution for Challenge 1: Transpose-on-Transfer (2)

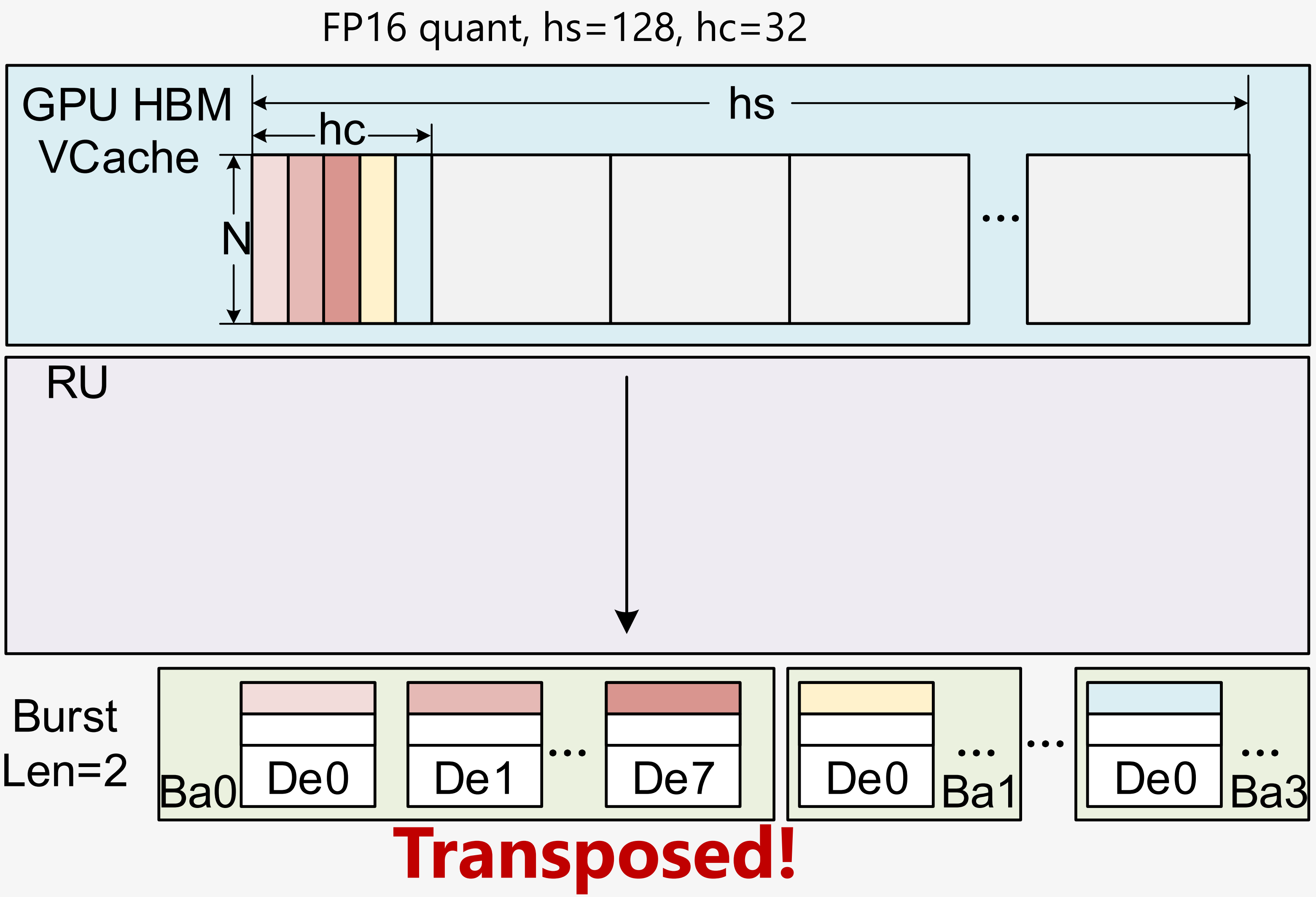


Solution for Challenge 1: Transpose-on-Transfer (2)

FP16 quant, hs=128, hc=32

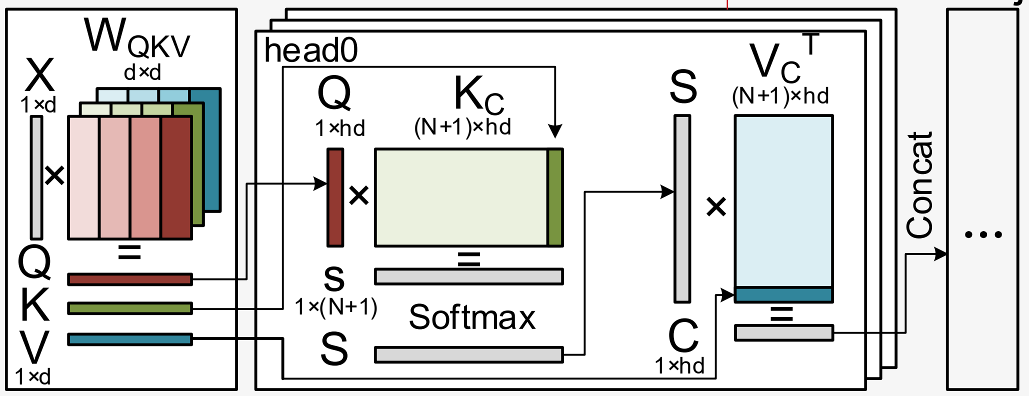


Solution for Challenge 1: Transpose-on-Transfer (2)



Communication Challenge 2 – Data Movement

Attention Layer Task Division on GPU



N=2048, d=4096,hc=32,hd=d/hc=128

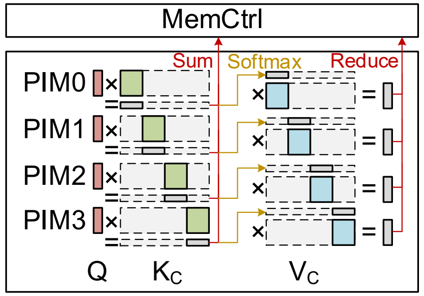
Divide Task based on ***Heads***

16~32 Heads

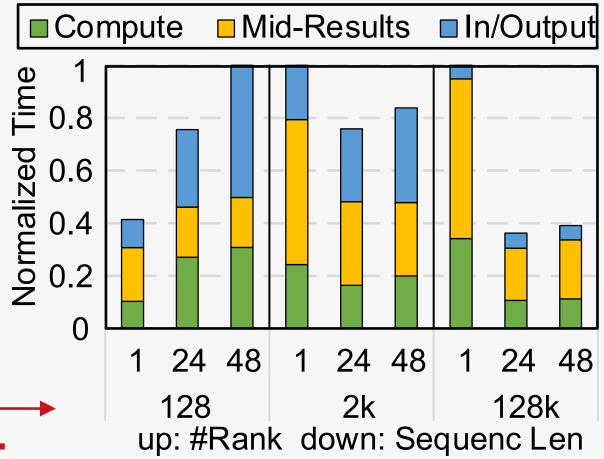
PIM Units Parallelism:

- A 8 Channel*4 Rank DDR5 Memory System:
  - 32 (Ranks) * 16 (Banks) -> 2048 (PIM Units)

Under-utilized!



Further tile the heads and fully utilize PIM parallelism



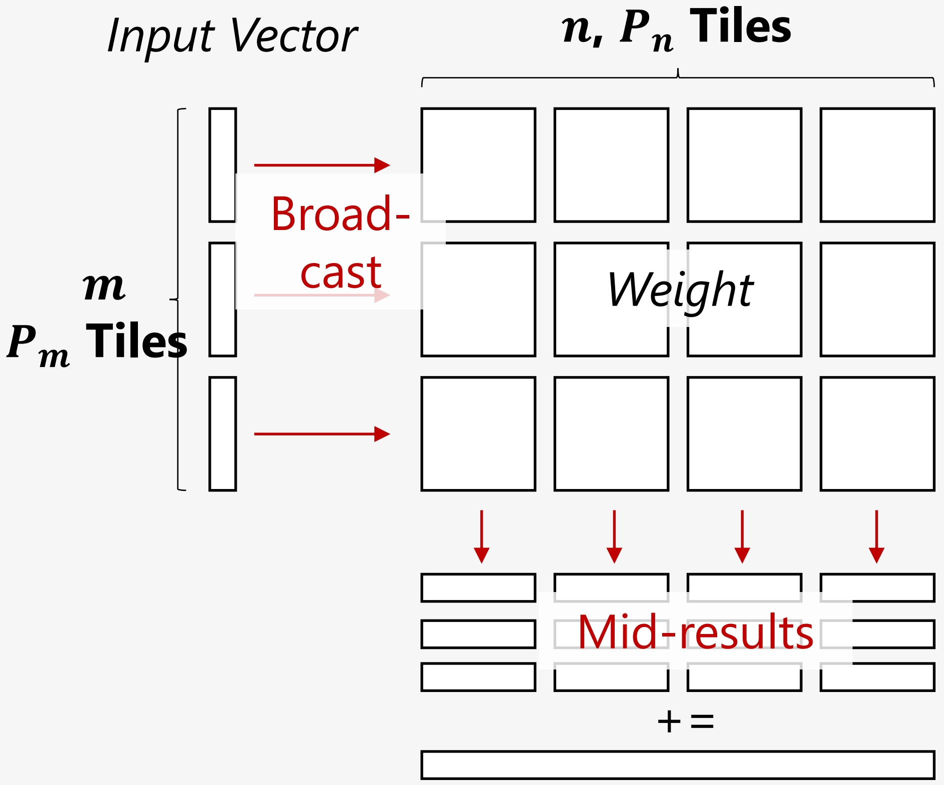
Leads to heavier mid-result movement overhead

Solution to Challenge 2: Dynamical Parallelism Scaling

上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

- Estimate the PIM Computing & Communication Overhead

- Linear Layer:



PIM Computation Time:

$$T_{comp} = \frac{mn}{P_m P_n \cdot BDW_{PIM}}$$

Communication Time:

Broadcast

Mid-Result (Intra-Channel)

Mid-Result (Inter-Channel)

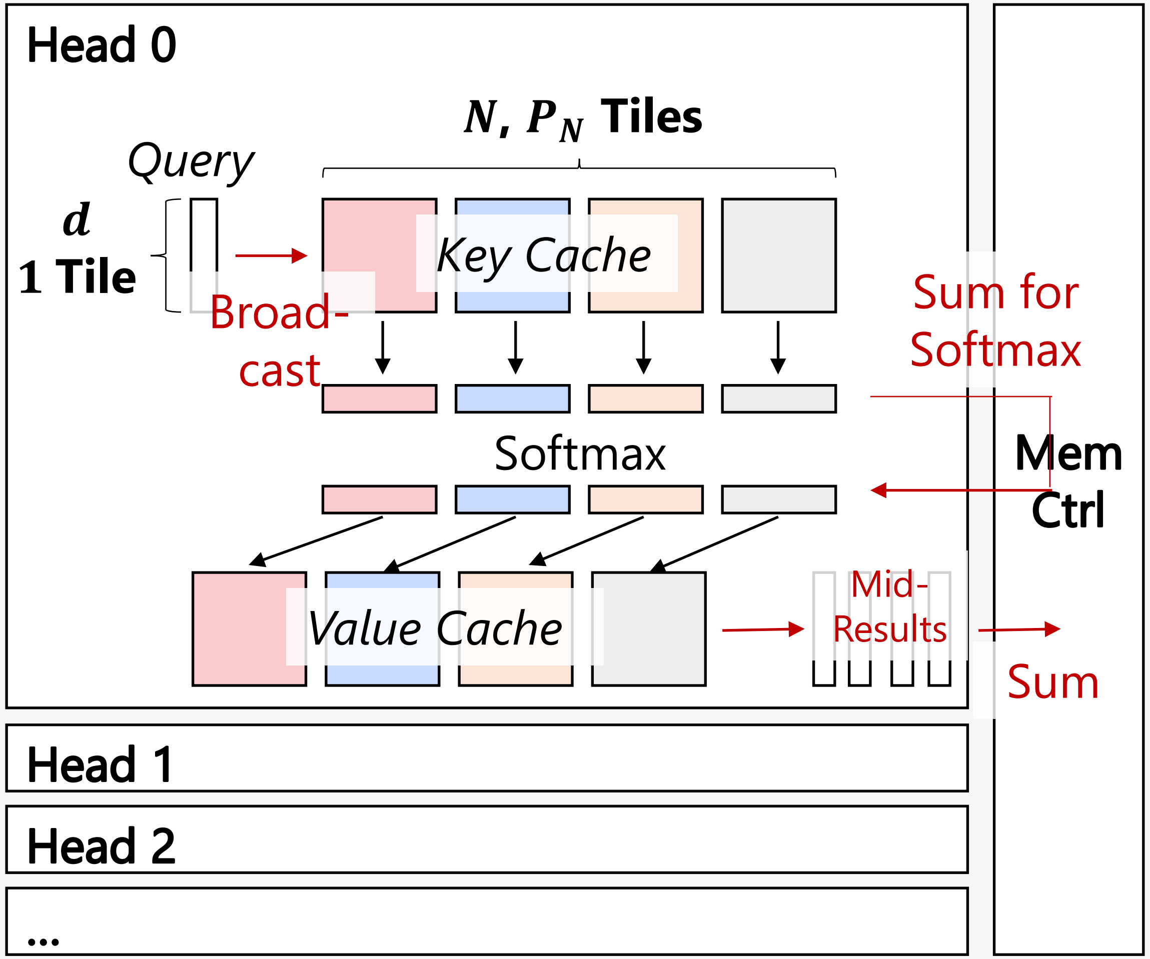
$$T_{comm} = \frac{mP_n + nP_m + n_c P_m}{n_c \cdot BDW_{DRAM}}$$

Bandwidth
(n_c: channel num)

Solution to Challenge 2: Dynamical Parallelism Scaling

上海交通大學 SHANGHAI JIAO TONG UNIVERSITY

Attention Layer:



PIM Computation Time:

$$T_{comp} = \frac{2dN}{P_N \cdot BDW_{PIM}}$$

Communication Time:

$$T_{comm} = \frac{\overset{\textbf{Broadcast}}{dP_N} + \overset{\textbf{Softmax Sum}}{2P_N} + \overset{\textbf{Mid-Result (Intra-Channel)}}{dP_N} + \overset{\textbf{Mid-Result (Inter-Channel)}}{n_c d}}{\underbrace{n_c \cdot BDW_{DRAM}}_{\textbf{Bandwidth } (n_c\textbf{: channel num})}}$$

Solution to Challenge 2: Dynamical Parallelism Scaling

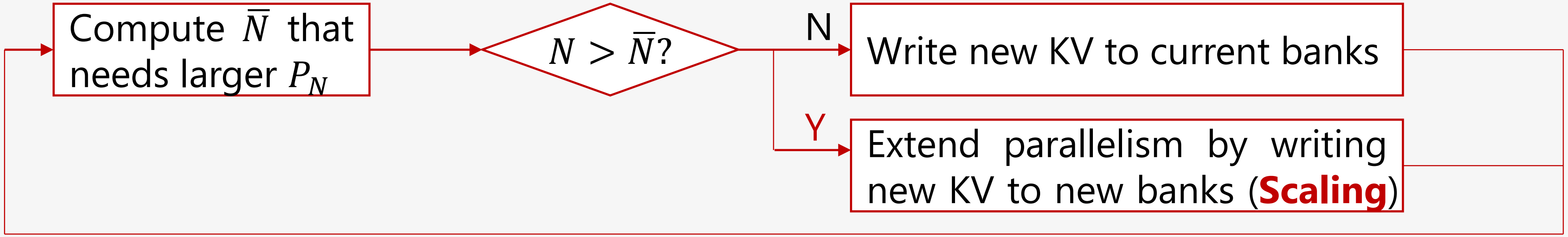
上海交通大学 SHANGHAI JIAO TONG UNIVERSITY

- PIM Computation Time: $T_{comp} = \dfrac{2dN}{P_N \cdot BDW_{PIM}}$
- Communication Time: $T_{comm} = \dfrac{dP_N + 2P_N + dP_N + n_c d}{n_c \cdot BDW_{DRAM}}$

N (#tokens in KV Cache) Grows during decoding!

Larger N needs larger P_N to balance computation & communication

Dynamical Parallelism Scaling:



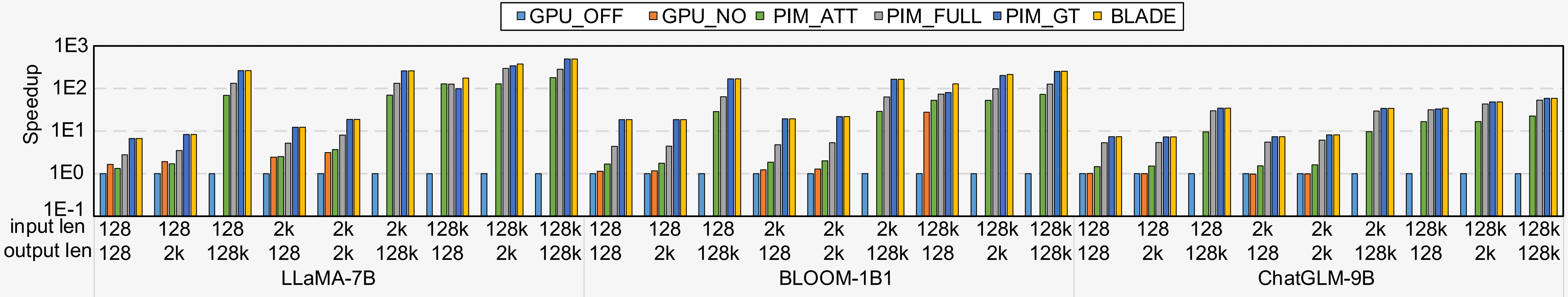
Experiments

Experimental Setup

- Benchmarks: **ChatGLM-9B** (GHA, G=2), **LLaMA-7B**, **BLOOM-1B1**
- GPU Baselines:
  - ***GPU_OFF***: Decoding with GPU, KV-Cache offloaded to CPU memory
  - ***GPU_NO***: Decoding with GPU, KV-Cache always in GPU memory
    - (Faster, but support shorter KV-Cache)
- PIM Baselines:
  - ***PIM_ATT***[ASPLOS'24]: Linear layer compute on GPU
  - ***PIM_GT***: V-cache transposed by GPU (Without Tech1 **Transpose-on-Transfer**)
  - ***PIM_FULL***: Always exploit full PIM parallelism (Without Tech2 **Dynamical Parallelism Scaling**)
- Simulation:
  - DRAM model: Ramulator2
  - Circuits: Synopsys DC + TSMC 90nm

System Configuration	
GPU	Nvidia V100, 112TFLOPS
DRAM	DDR4, 12 Channel*4 Ranks
PIM	32GOPS, 64 per Rank

Results (1) – Overall Performance



3 KV Cache lengths: 128, 2k, 128k

BLADE:

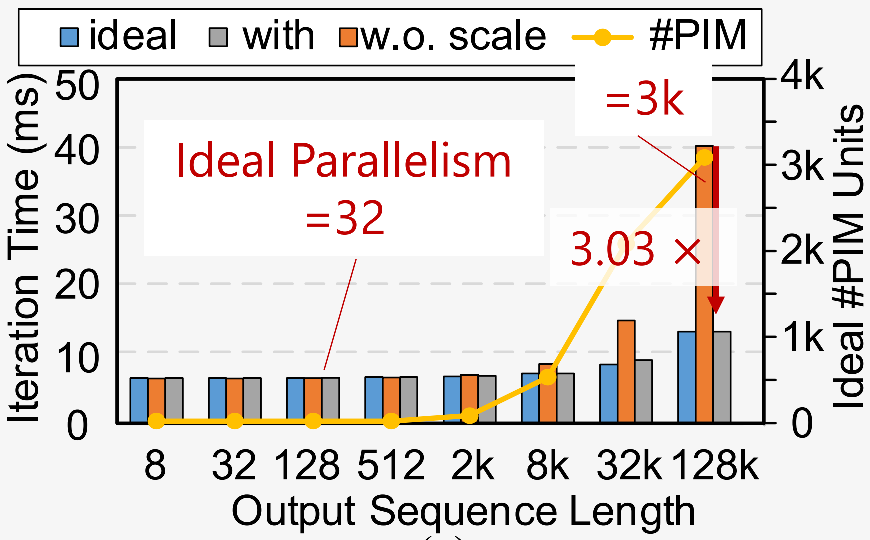
4.93 × to ***PIM_ATT*** (Linear Layers are on GPU)

105.7 × to ***GPU_OFF***

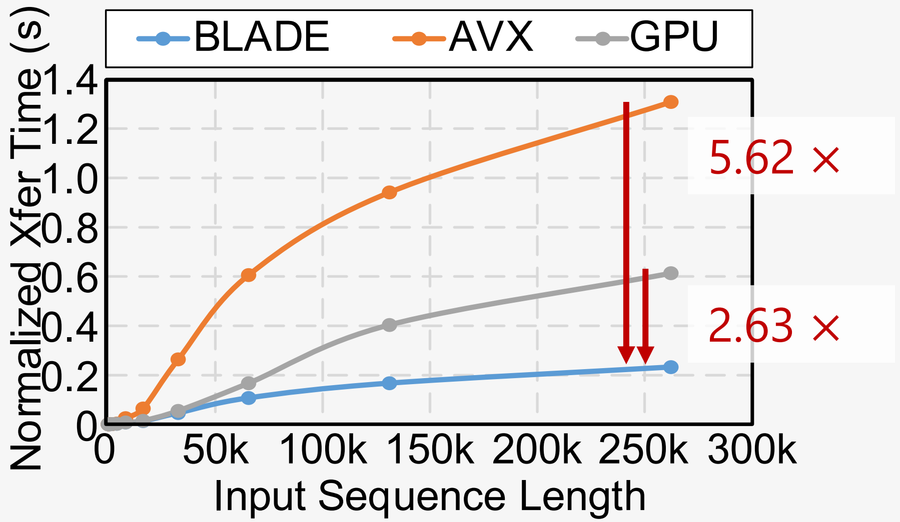
2.09 × to *PIM_FULL* (w.o. Tech 2 Dynamical Parallelism Scaling)

1.54 × to *PIM_GT* (w.o. Tech 1 Transpose on Transfer)

Results (2)



(a)



(b)

Dynamical Parallelism Scaling:

Baseline: Always use the initial parallelism

3.03 × for 128k KV Cache

Transpose-on-Transfer:

2.63 × to **GPU**-based Transpose
5.62 × to **CPU**-based Transpose (AVX Instructions)

Summary

Context: Large Language Model (LLM) inference, **Prefill-Decoding Disaggregation**

- Prefill->GPU; Decoding -> DRAM-based **Processing-in-Memory (PIM)**

Challenges on Communication & Solutions:

- Challenge 1: High KV Cache Transpose Latency During “GPU->PIM” KV Cache Transfer
- Solution 1: **Transpose on Transfer Method**
- Challenge 2: Static PIM Parallelism Fails to Balance Communication & Computation
- Solution 2: **Dynamical Parallelism Scaling Method**

Results:

- Transpose on Transfer 1.54x; Dynamical Parallelism Scaling 2.09x

上海交通大學

SHANGHAI JIAO TONG UNIVERSITY

Thank You!

饮 水 思 源　爱 国 荣 校