CREDIT: Cost-guided Reduction-reuse with Efficient DSMEM Inter-CTA Tiling

Zhengxiong Li, Tsung-Wei Huang, Umit Ogras

University of Wisconsin–Madison

IEEE High Performance Extreme Computing Virtual Conference (HPEC)

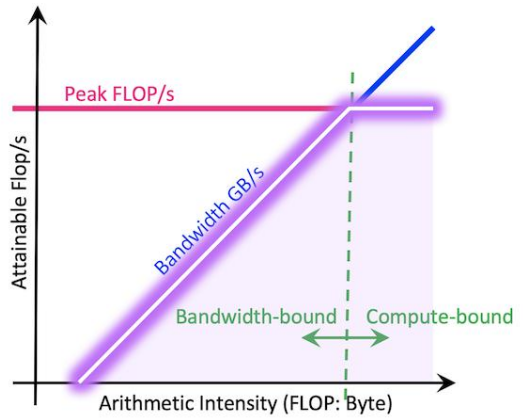
September 16, 2026

Introduction – Memory Wall

- **Memory Wall**
  - Modern GPUs' peak performance has grown faster than off-chip memory bandwidth
  - Architecture evolution:
    - Volta → Ampere → Hopper → Blackwell
  - For a given kernel,
    - $Perf. = \min(P_{peak\ compute}, AI \times BW_{memory})$

TABLE I: Compute-to-memory scaling in representative NVIDIA GPUs

GPU	Arch.	Peak Perf. (TFLOP/s)	Mem. BW (TB/s)	Roofline Ridge (FLOPs/byte)
V100 SXM	Volta	125	0.900	**139**
A100 SXM	Ampere	312	2.039	**153**
H100 SXM	Hopper	990	3.350	**295**
B200 SXM	Blackwell	2250	8.000	**281**
RTX 5090	Blackwell	419	1.792	**234**

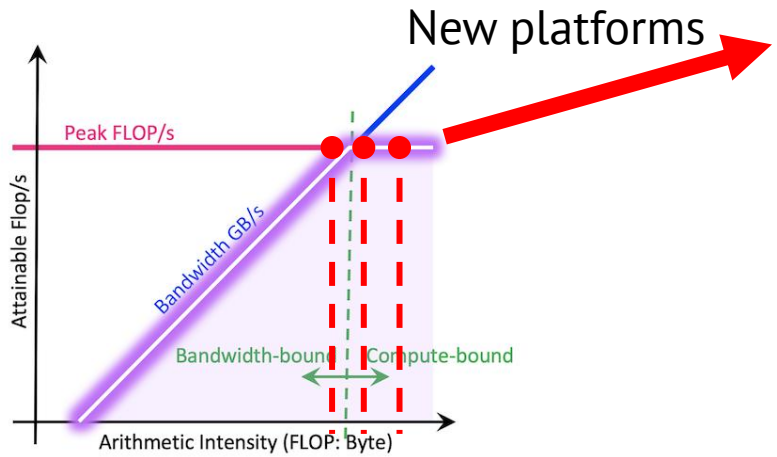


Introduction – Memory Wall

- **Memory Wall**
  - In ideal cases, the arithmetic intensity of a workload should stay around the ridge point
  - When switching to a newer platform, peak performance becomes higher, and its ridge point moves to the right
  - A compute-bounded workload may become memory-bounded, if we don't optimize it on new platforms

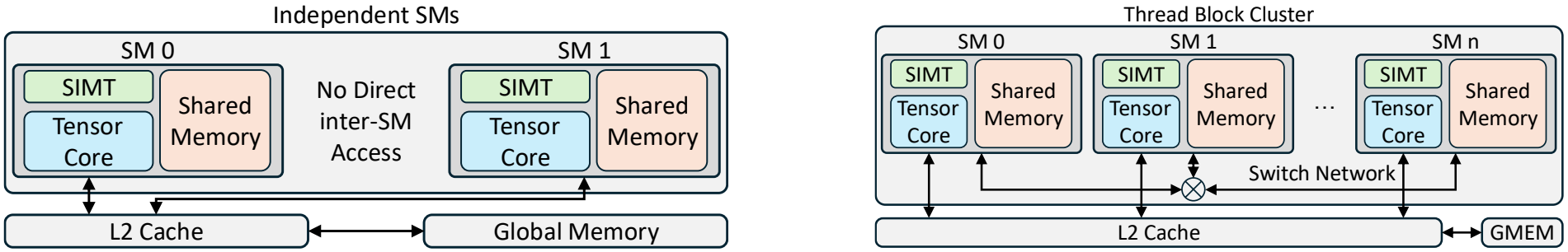
TABLE I: Compute-to-memory scaling in representative NVIDIA GPUs

GPU	Arch.	Peak Perf. (TFLOP/s)	Mem. BW (TB/s)	Roofline Ridge (FLOPs/byte)
V100 SXM	Volta	125	0.900	**139**
A100 SXM	Ampere	312	2.039	**153**
H100 SXM	Hopper	990	3.350	**295**
B200 SXM	Blackwell	2250	8.000	**281**
RTX 5090	Blackwell	419	1.792	**234**



Background – DSMEM

- **DSMEM**
  - Starting from Hopper architecture, NVIDIA introduced thread block cluster and distributed shared memory (DSMEM)
  - A cluster is a group of thread blocks that are co-scheduled and can synchronize as a unit
  - DSMEM allows these blocks to directly load, store, and perform atomic operations on one another's shared memory
  - However, there is no free lunch



Shared-memory access with and without a thread block cluster. DSMEM lets a CTA address allocations owned by peer CTAs through the inter-SM interconnect; each allocation remains physically local to its owner

CREDIT – Profiling

- **DSMEM Profiling**
  - We profiled the performance of DSMEM with CUDA microbenchmarks
  - Two key takeaways:
    - A remote load has about 6.4× the latency and 4.4–5.4× lower throughput than a local read → moves only compact partial results between CTAs and reads them locally after synchronization
    - Synchronization and remote-store costs differ substantially between the two GPUs → cost model

Metric	Latency	Bandwidth	Latency	Bandwidth
	RTX 5090		H100	
L1	69	~	41	~
L2	361	~	263	~
GMEM	915	1.5 TB/s	479	3.35 TB/s

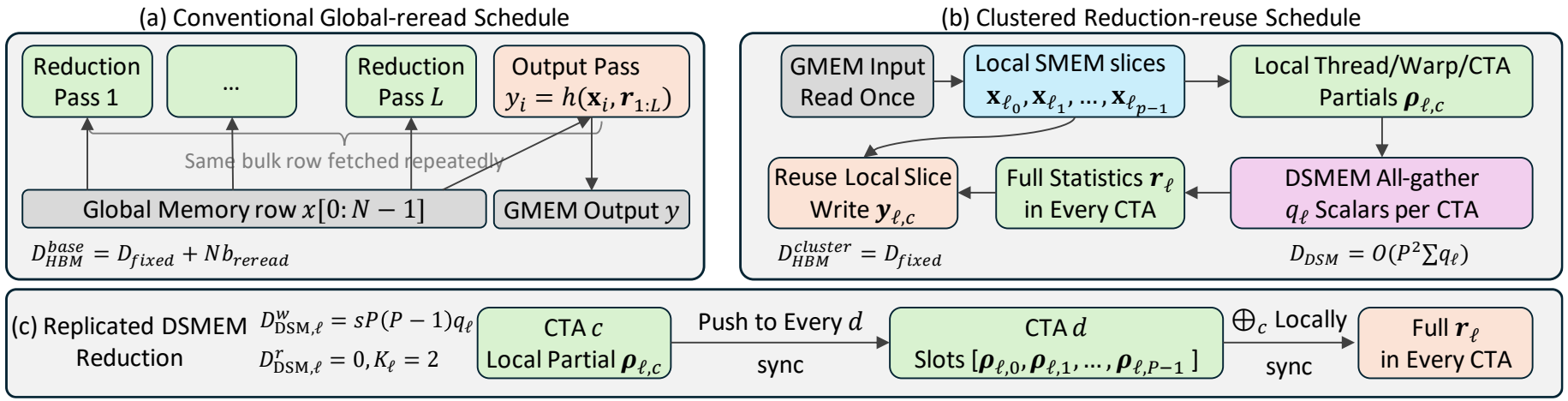
Latency in cycles

TABLE II: Profiling on RTX 5090 and H100

GPU	Latency (cycles)			Rate (byte/cycle)			
	Local load	Remote load	Sync	Local read	Remote read	Local store	Remote store
RTX 5090	34	216	404	20.89	3.84	33.01	2.14
H100	30	191	851	20.89	4.77	27.67	21.37

CREDIT – Cost Model

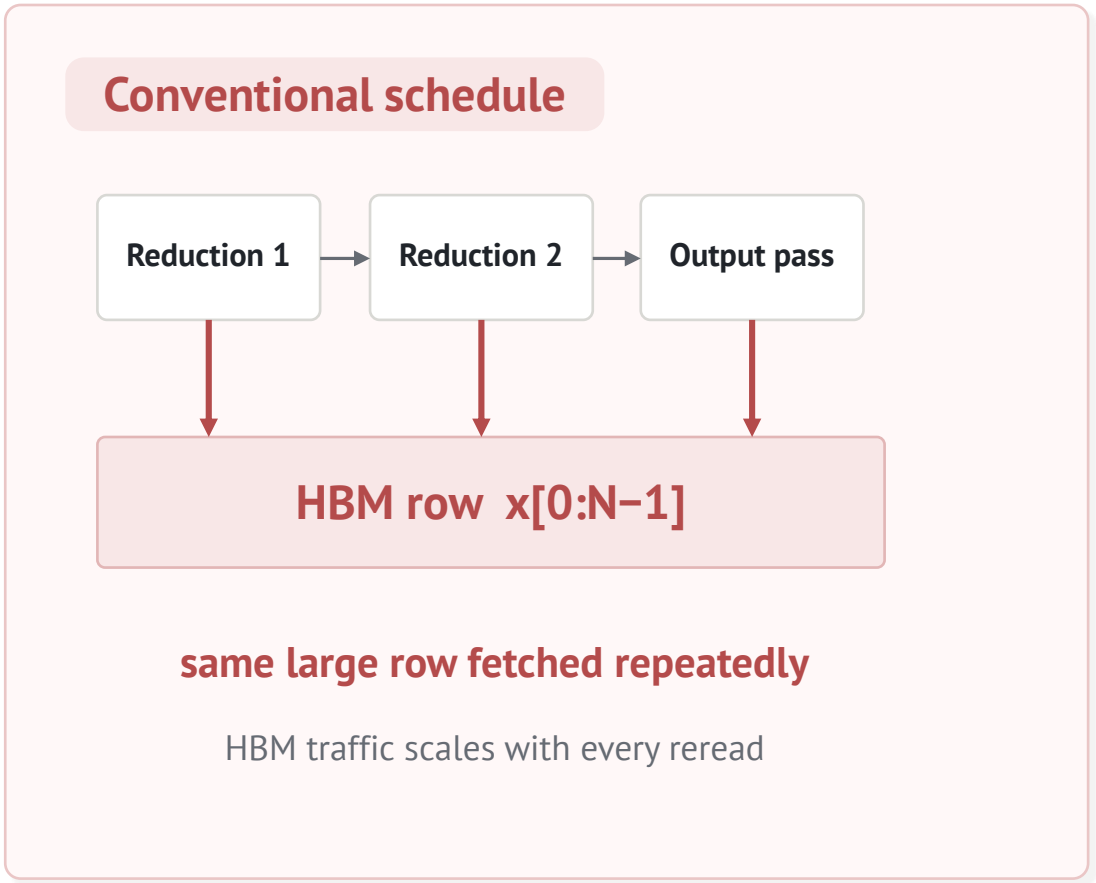
- **Cost Model**
  - We want to know, for a given workload and shape, whether it can benefit from DSMEM

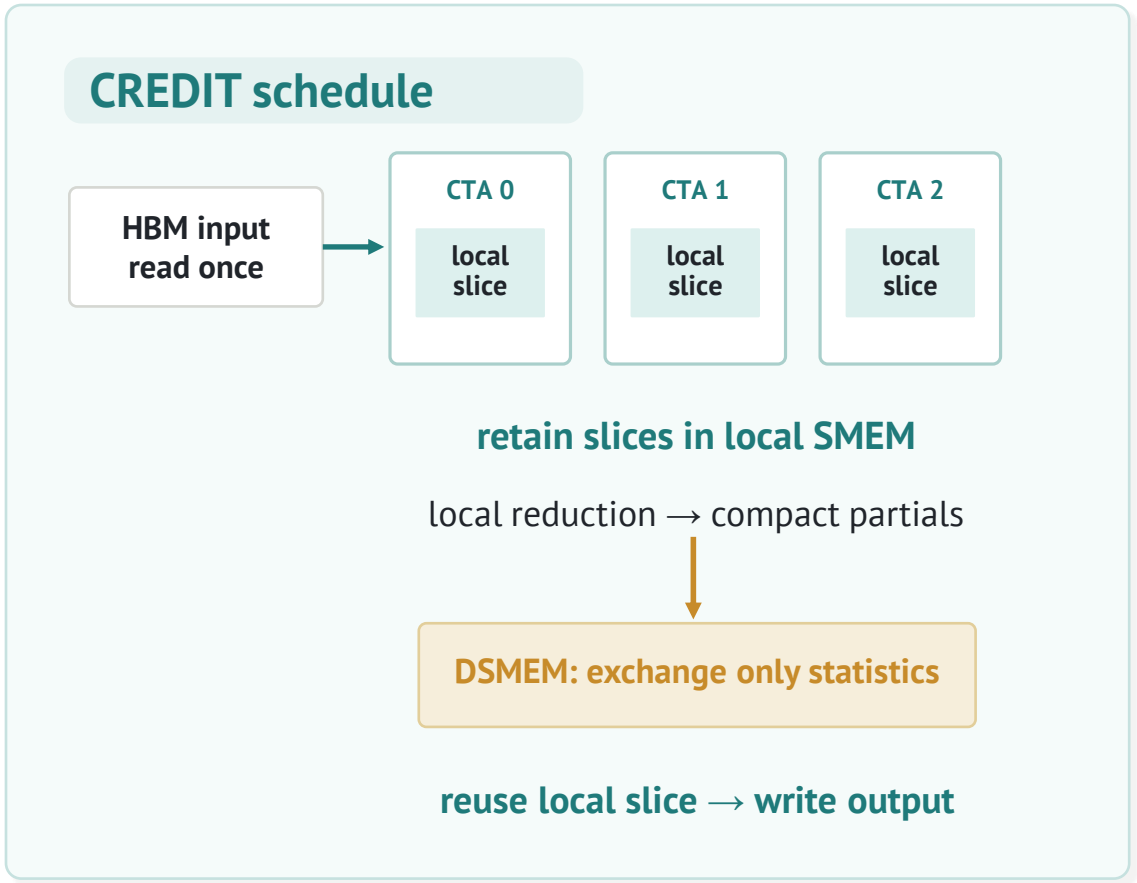


CREDIT transformation for reduction-reuse kernels. (a) Conventional passes reread bulk rows. (b) A cluster retains owner-local slices and exchanges scalar partials. (c) Each CTA pushes partials to every peer, which combines its local slots after synchronization

CREDIT – Cost Model

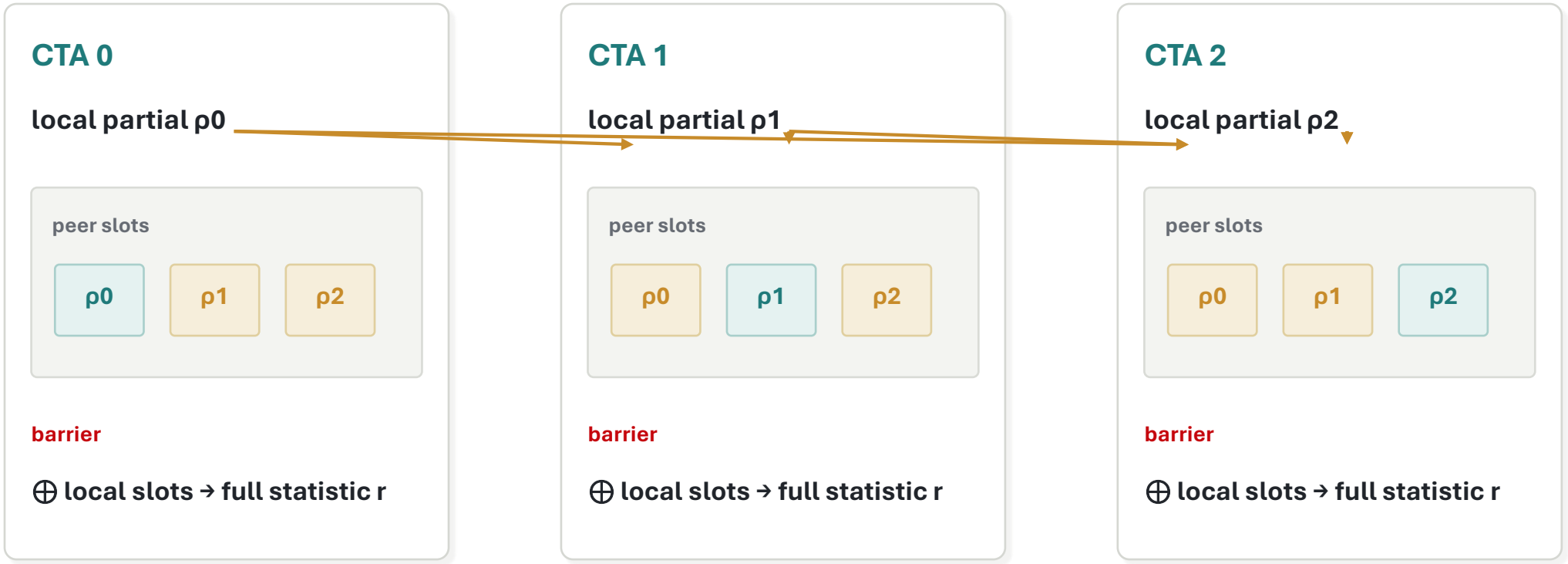
- **Cost Model**
  - Target pattern: wide reduction-reuse kernels that reread the same bulk row across passes





CREDIT – Cost Model

- **Replicated push**
  - Remote stores, then local reads



Result: DSMEM carries $O(P^2 \cdot \#statistics)$ compact stores; the combining reads are owner-local.

CREDIT – Cost Model

- **Use DSMEM only when saved HBM time exceeds its cost**
  - CREDIT is a screening model, not a cycle-accurate runtime emulator

$$\Delta T \approx T_{save} - T_{ctrl} - T_{replay} - T_{deposit} - T_{DSMEM}$$

profitable when $\Delta T > 0$

BENEFIT

Avoided source rereads

Value the eliminated bytes at the baseline's achieved rate — not peak bandwidth.

COSTS

Cluster overheads

Synchronization + scheduling / occupancy + local replay + DSMEM transport.

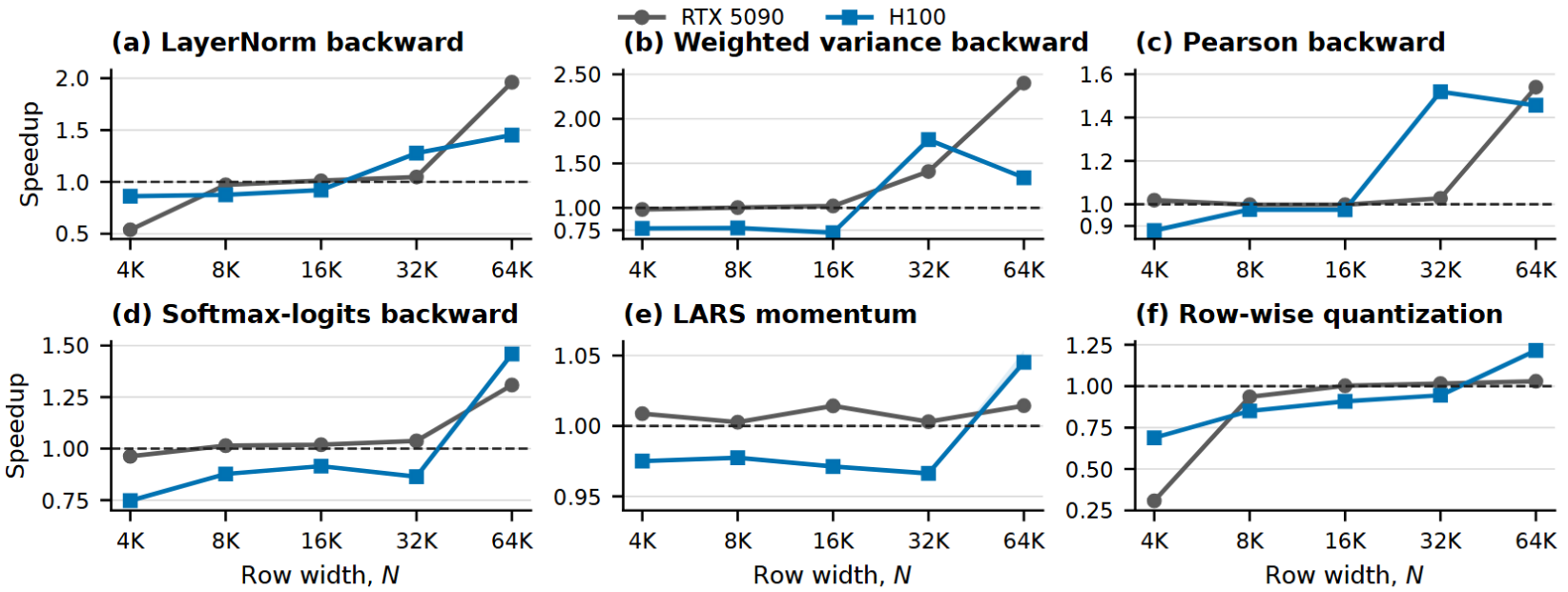
Inputs: static workload features | one baseline timing | hardware profiling → generate CREDIT or keep baseline

Evaluations

- **Evaluations**
  - NVIDIA RTX 5090 and H100, with CUDA 13.0, Pytorch 2.11 and Triton 3.6.0
  - Baselines: torch.compile, Triton and non-cluster native CUDA
    - CREDIT beats the best baseline as problem size becomes larger
    - Average speedups of 1.466× and 1.318×, respectively

TABLE IV: Geometric-mean CREDIT speedup at $N = 65536$ over each non-DSMEM baseline and the fastest baseline at each workload.

GPU	`torch.compile`	Triton	CUDA	Best
RTX 5090	1.617×	1.937×	1.980×	1.466×
H100	1.469×	1.525×	1.533×	1.318×



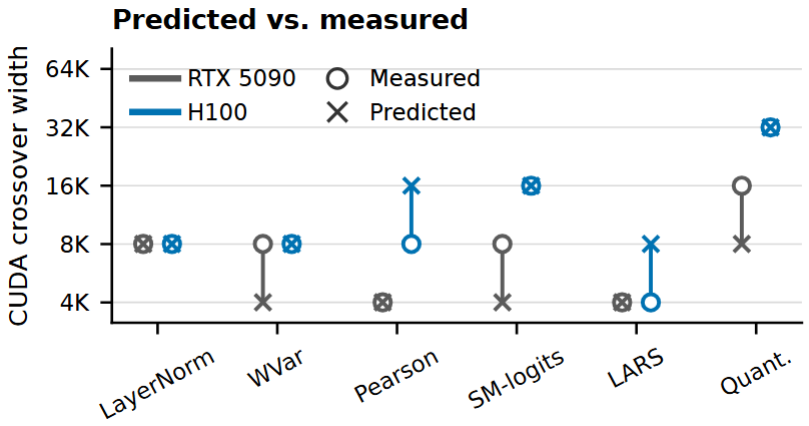
CREDIT speedups over the fastest baseline across workloads and shapes

Evaluations

- **Cost Model Validation**
  - We obtain local-SMEM, and DSMEM rates from profiling; the work-free control supplies T_{ctrl} for each (M, N, P); the non-DSMEM kernel supplies T_B ; and static analysis supplies $b_B, b_{keep}, b_{reread}$
  - Tested with 60 profitability pairs: five input size, six workloads, and two GPUs
  - Reaches 90.0% accuracy on RTX 5090, 93.3% on H100, and 91.7% overall

TABLE V: Profitability-pair accuracy across six workloads and five row widths per GPU.

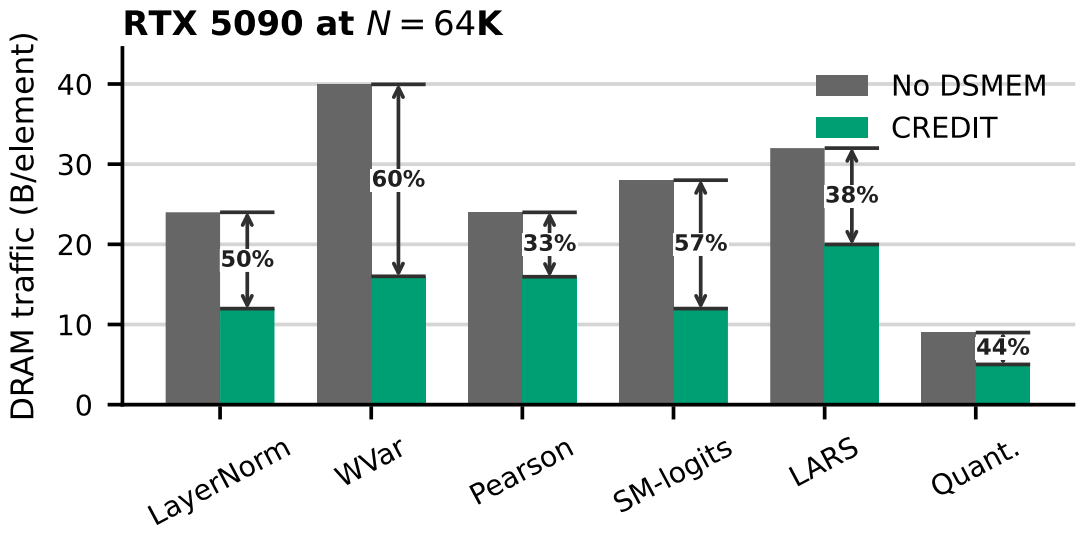
Model	RTX 5090	H100	Overall
Peak bandwidth + serialized staging	27/30	16/30	43/60
CREDIT: achieved bandwidth + overlap	27/30	28/30	55/60



Crosses and open circles show the predicted and measured smallest width at which DSMEM beats matched non-DSMEM CUDA

Evaluations

- **Memory Traffic Validation**
  - DSMEM reduces measured RTX 5090 DRAM traffic by 33–60%
  - Owner-local staging removes repeated bulk rereads; output traffic stays unchanged
  - Traffic reduction alone does not predict speedup — overhead and baseline efficiency still matter



DRAM traffic reduction, collected by NCU profiler

Takeaways

1. **DSMEM is selective, not universal**
   Target wide reduction-reuse kernels with recoverable reread traffic

2. **Keep bulk data local**
   Exchange compact partial statistics and combine from owner-local SMEM

3. **Predict before generating**
   Use achieved baseline bandwidth + hardware profiling to screen profitability

1.466×	1.318×	91.7%
RTX 5090	H100	Modal Accuracy

github.com/zhengxiongli08/CREDIT

Thank you all! Questions?