Coruscant: Co-Designing GPU Kernel and Sparse Tensor Core to Advocate Unstructured Sparsity in Efficient LLM Inference

Donghyeon Joo[1], Helya Hosseini[1], Ramyad Hadidi[2], Bahar Asgari[1]

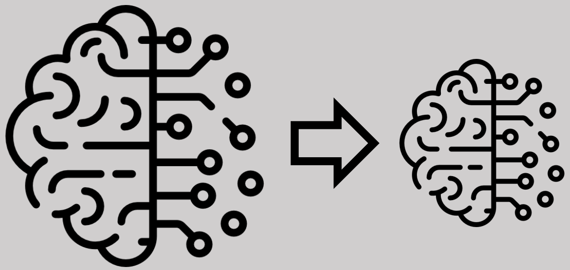
[1]University of Maryland, College Park

[2]d-Matrix

LLMs are huge and memory-hungry

Various model compression techniques are used

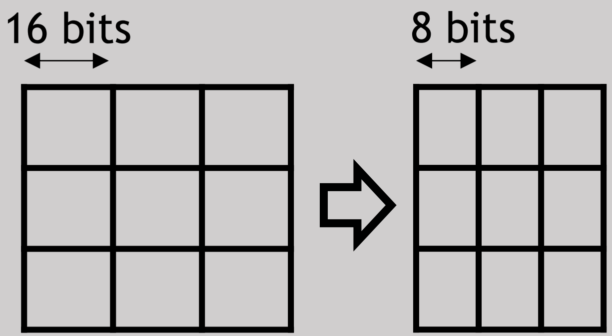
Distillation



Small model learns from larger model

DeepSeek-R1 [2025]

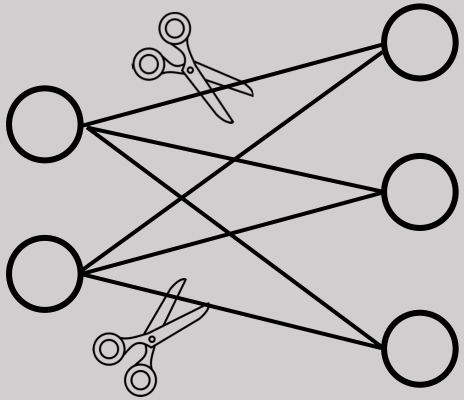
Quantization



Reduce model bit-width

gpt-oss [2025]

Pruning



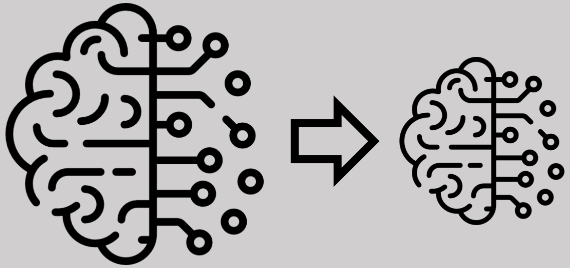
Reduce neural connections

?

LLMs are huge and memory-hungry

Various model compression techniques are used

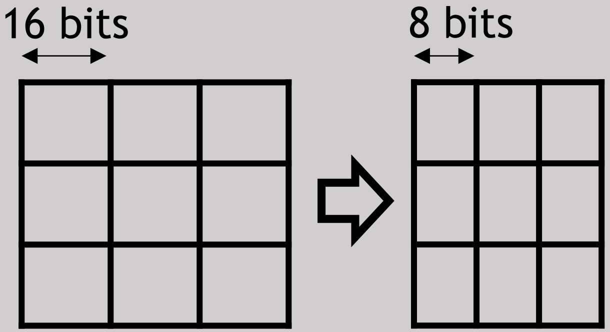
Distillation



Small model learns from larger model

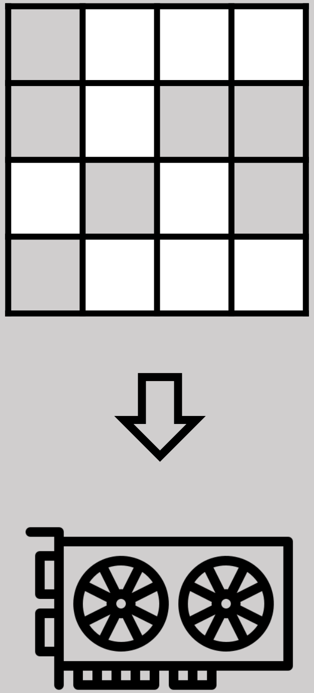
DeepSeek-R1 [2025]

Quantization



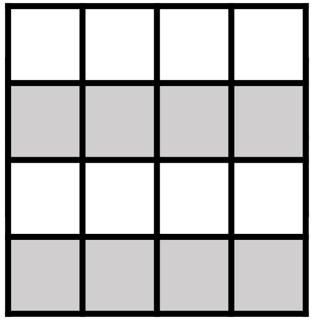
Reduce model bit-width

gpt-oss [2025]

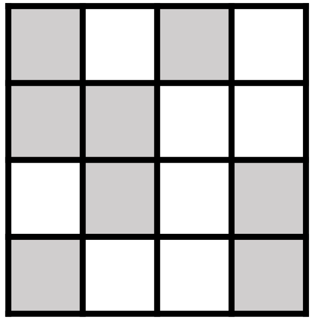


Tradeoff in Model Accuracy and Efficiency

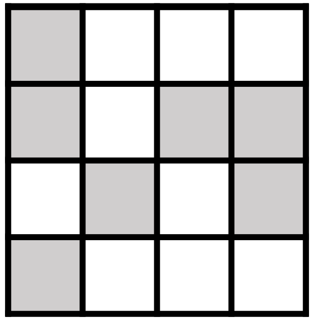
Structured Sparsity



Semi-Structured Sparsity

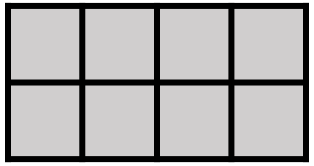


Unstructured Sparsity



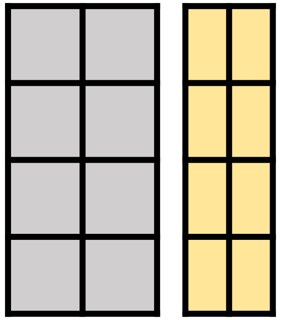
Tradeoff in Model Accuracy and Efficiency

Structured Sparsity



Smaller dense matrix

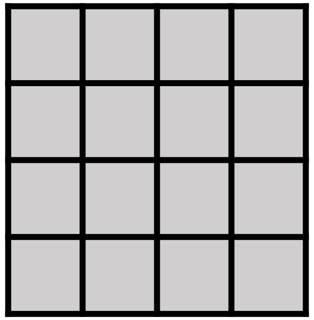
Semi-Structured Sparsity



position metadata

Constant number of non-zeros

Unstructured Sparsity



Compute as dense

Efficient Hardware Mapping

Accuracy Retention

Tradeoff in Model Accuracy and Efficiency

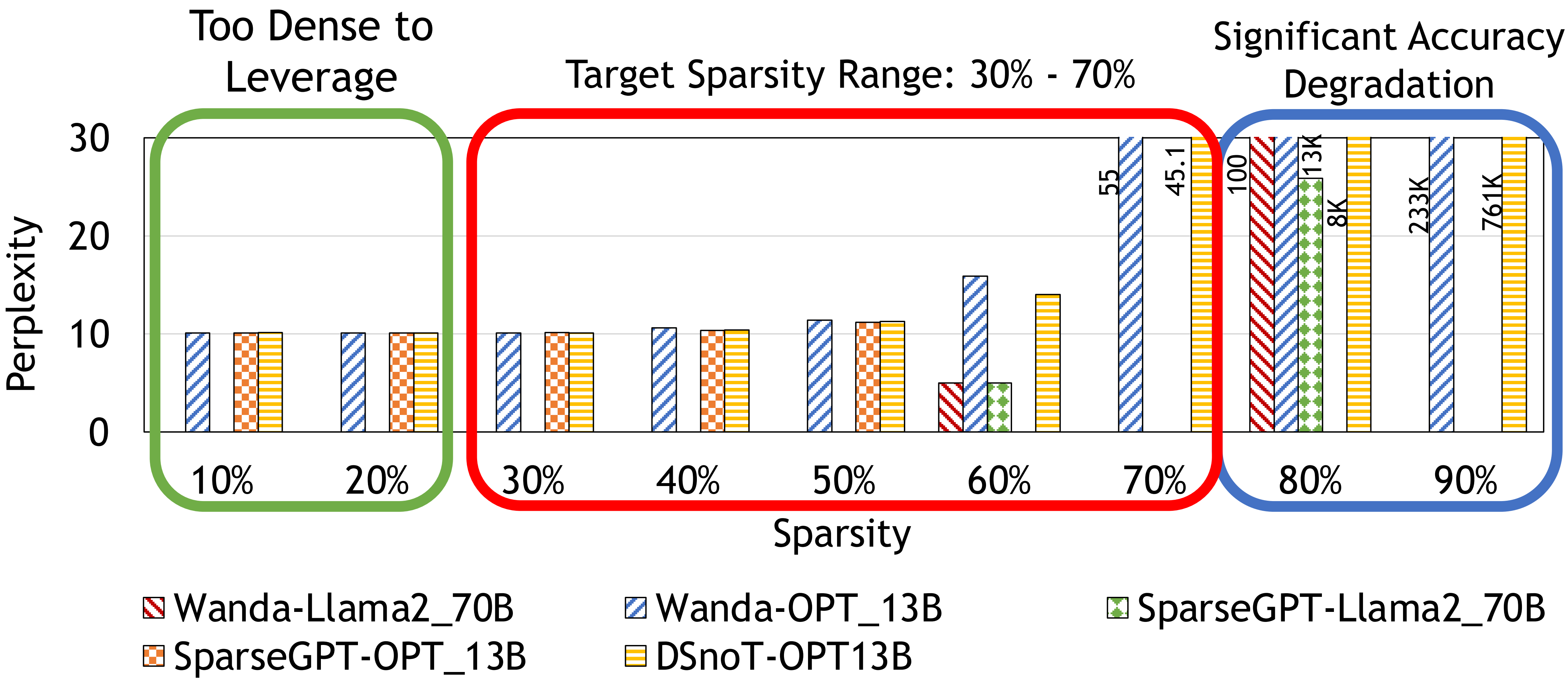
Methods† (pattern)	LLM	Perplexity Diff. ‡
SparseGPT (unstructured)	OPT-175B	-0.14
SparseGPT (2:4)	OPT-175B	0.39
Wanda (unstructured)	Llama-65B	1.01
Wanda (2:4)	Llama-65B	2.69

Perplexity on WikiText-2.
Perplexity difference from dense model, lower the better

	25% (3:4)	37.5% (3:8)	50% (2:4)	62.5 % (5:8)	75% (1:4)
Semi-Structured	84.22	75.70	6.09	0.06	0.00
Unstructured	**88.40**	**84.43**	**42.81**	**2.21**	**0.43**

TriviaQA accuracy on Llama-2 7B
Pruned to each sparsity with Wanda.

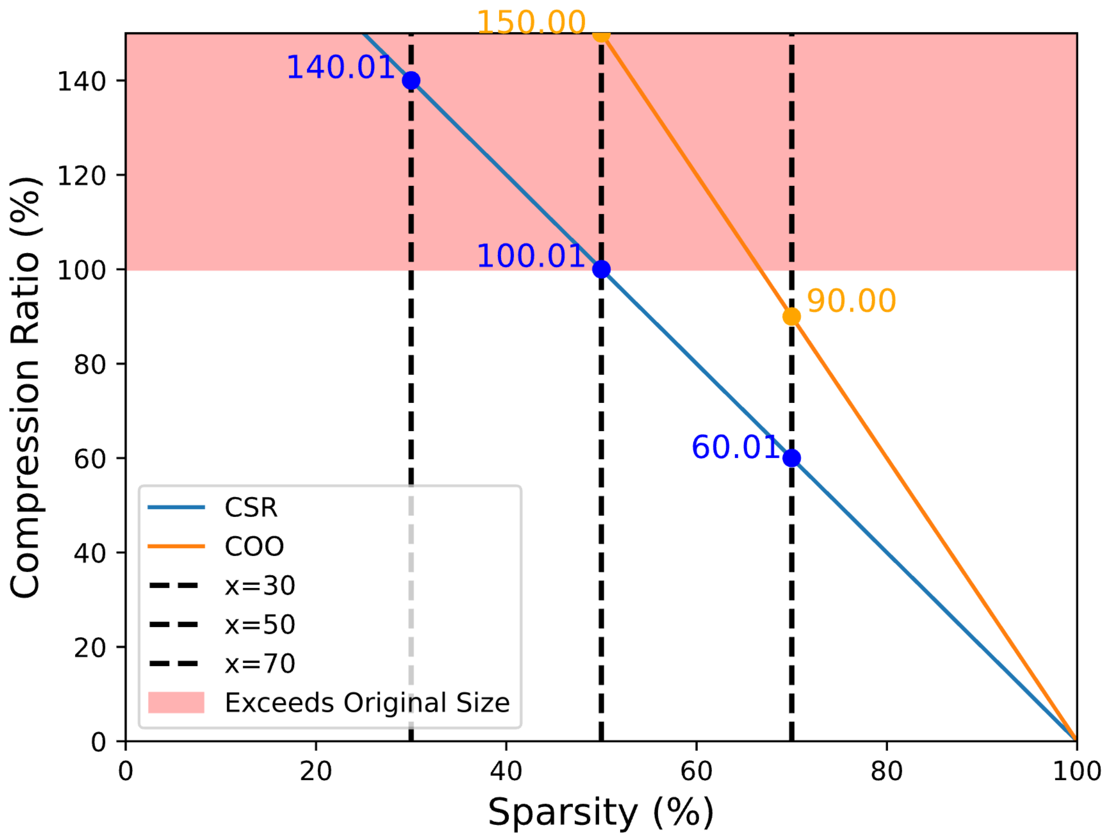
Effective Sparsity Range in LLM Pruning



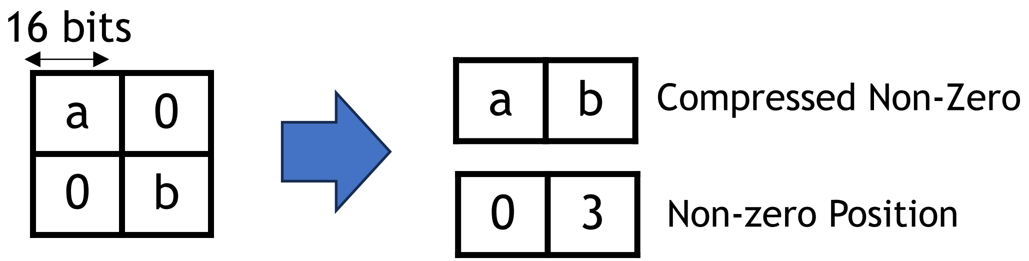
d-Matrix

DEPARTMENT OF COMPUTER SCIENCE

Efficient Unstructured Sparsity in Hardware

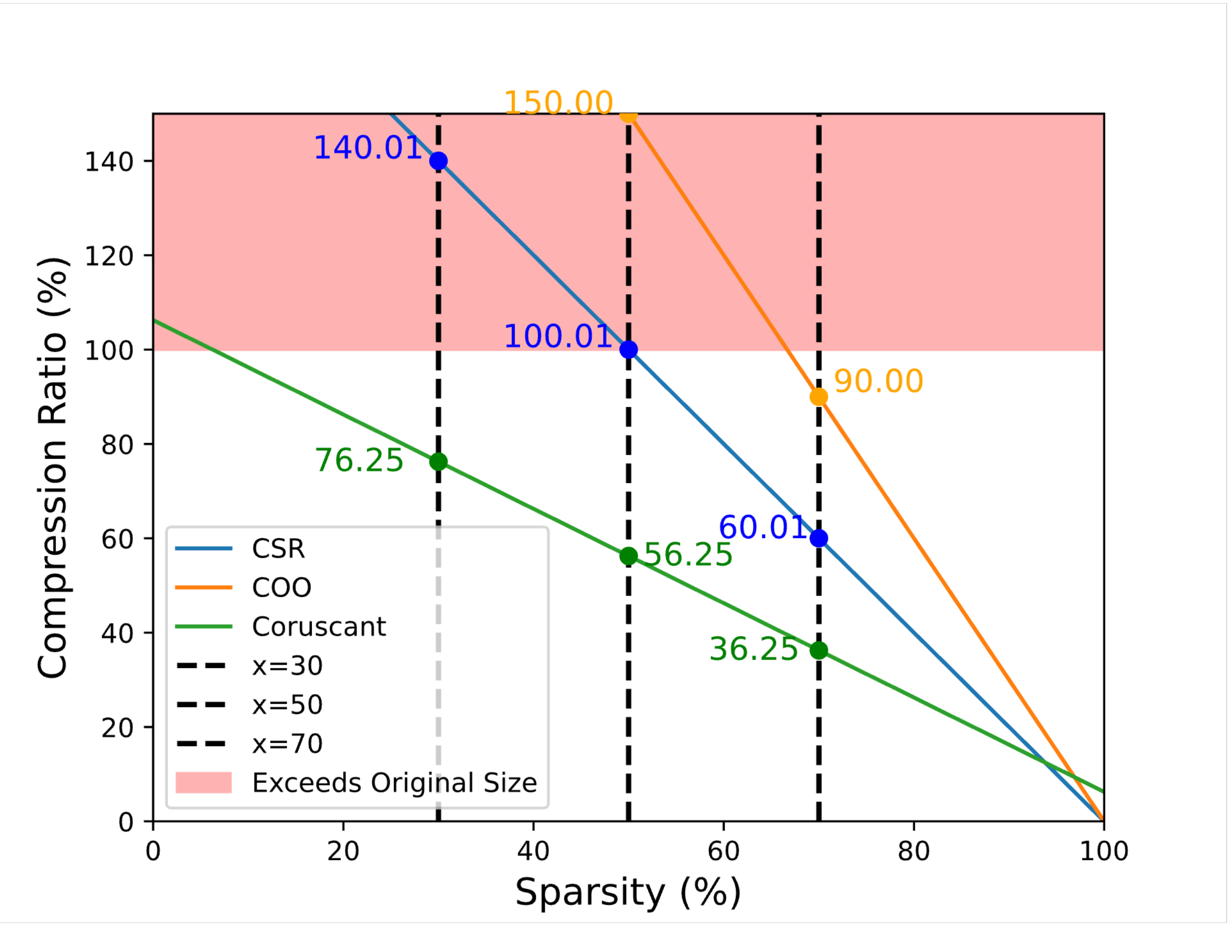


- We target a moderately sparse region.
- Sparse formats with numerical non-zero positions fail to compress our target.



Flash-LLM sparse format at 50% sparsity

Efficient Unstructured Sparsity in Hardware

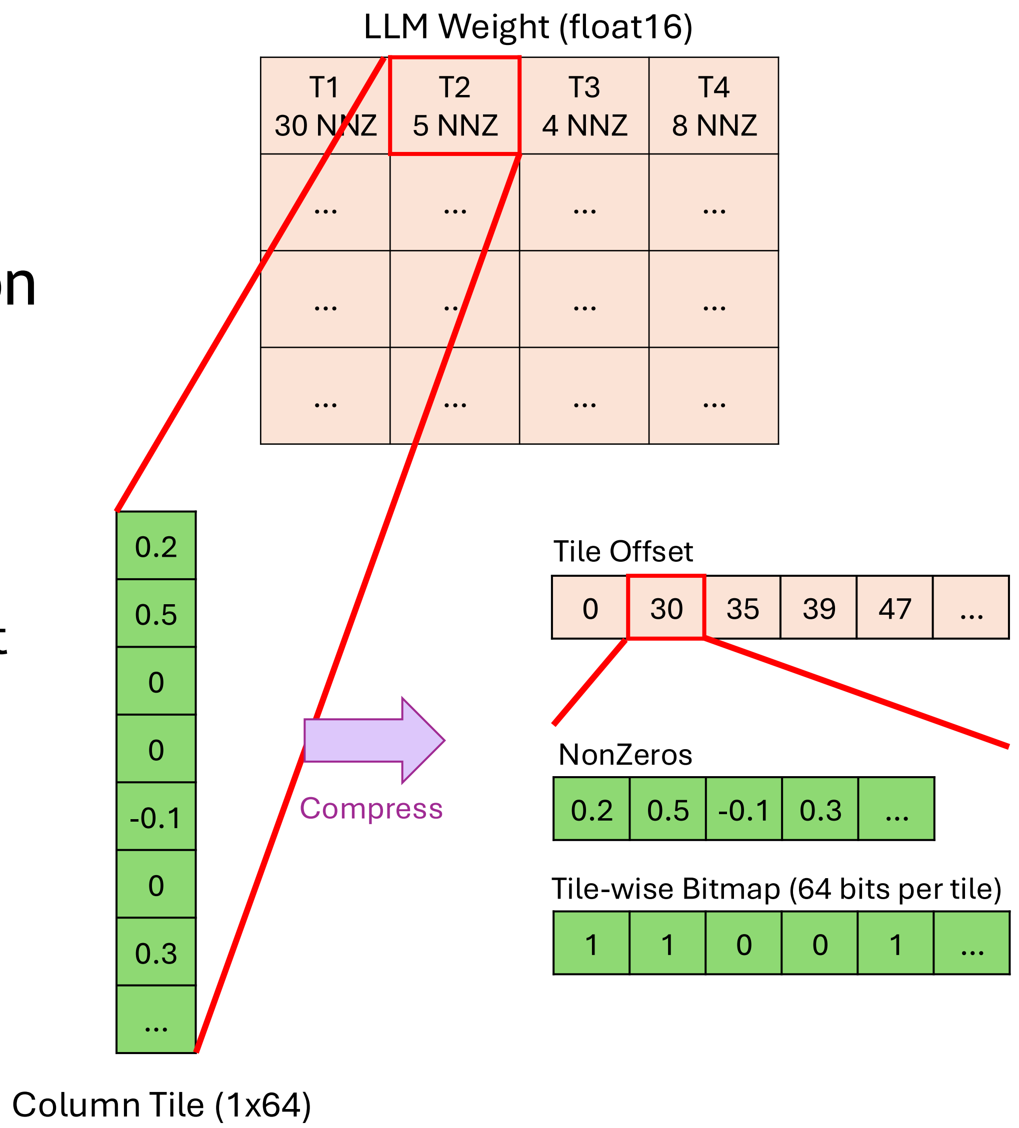


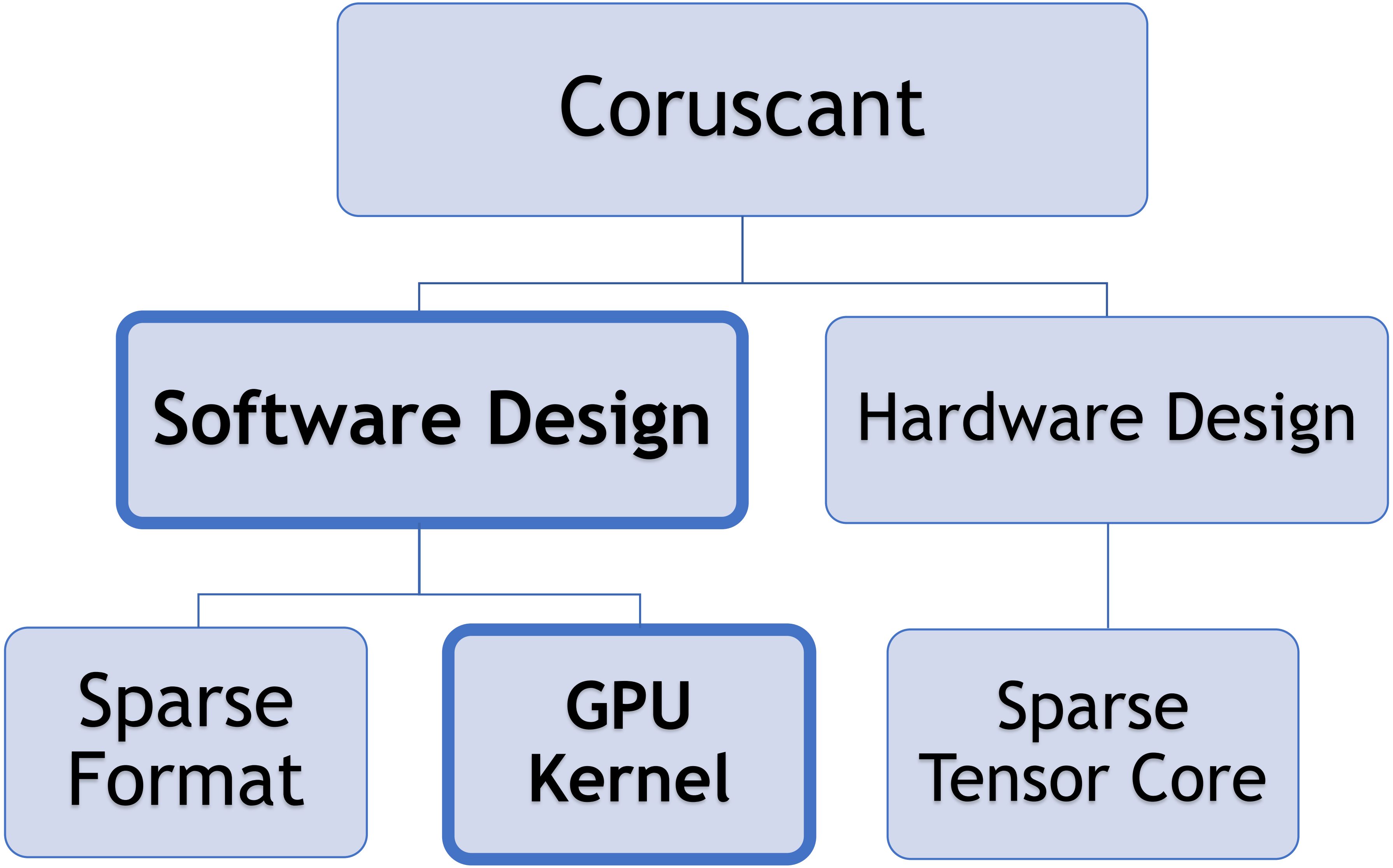
- We target a moderately sparse region.
- Sparse formats with numerical non-zero positions fail to compress our target.
- Bitmap-based representation is the only plausible solution for compression.

- Coruscant
  - **Software Design**
    - **Sparse Format**
    - GPU Kernel
  - Hardware Design
    - Sparse Tensor Core

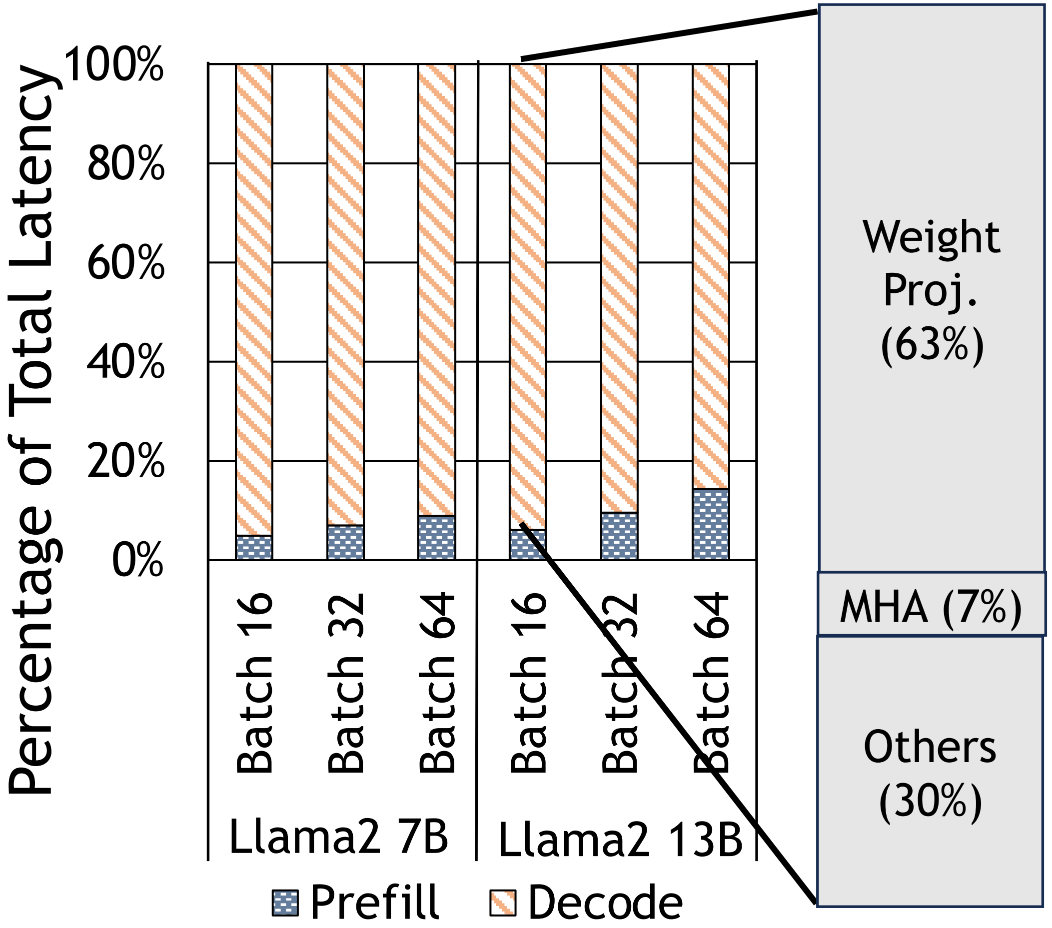
Coruscant Sparse Format

- Bitmap-based non-zero position representation
- GPU Kernel and Architecture considerations:
  - Each warp thread assigned to two tiles
  - Column-wise tiling to avoid shared memory bank conflict
- Maximal Compression Benefits:
  - Accelerated memory-bound SpMM
  - Reduced global memory consumption




Coruscant
Software Design
Hardware Design
Sparse Format
GPU Kernel
Sparse Tensor Core

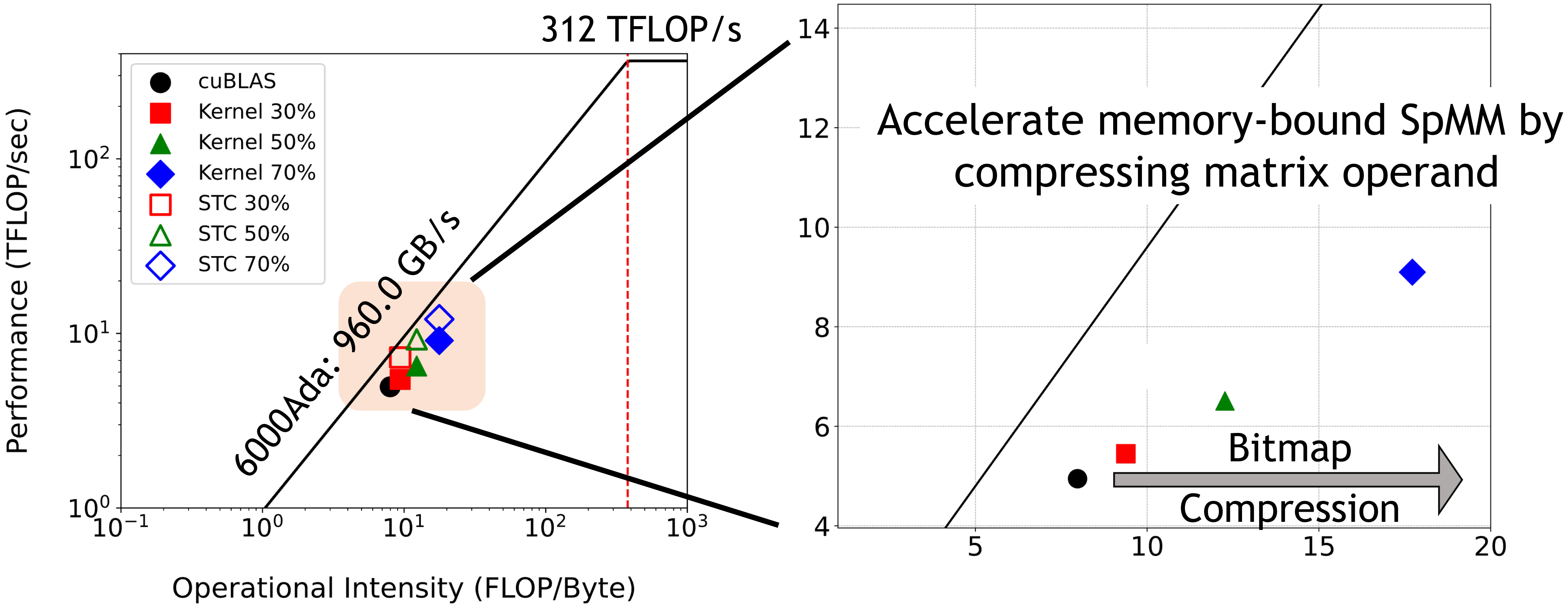
Coruscant GPU Kernel - Target Operation



Latency Comparison of Prefill and Decode

- Inference is dominated by decode phase.
- Decode phase is dominated by weight projection SpMM.
- Due to small batch dimension, weight projection SpMM is memory bound.

Coruscant GPU Kernel - Target Operation



Coruscant GPU Kernel - Load-Compute Pipeline

- Objective: Keep the data transfer from GMEM to GPU processor compressed.
- Phase 1: Matrix tiles are transferred to processor registers as-compressed, then decompressed to shared memory.

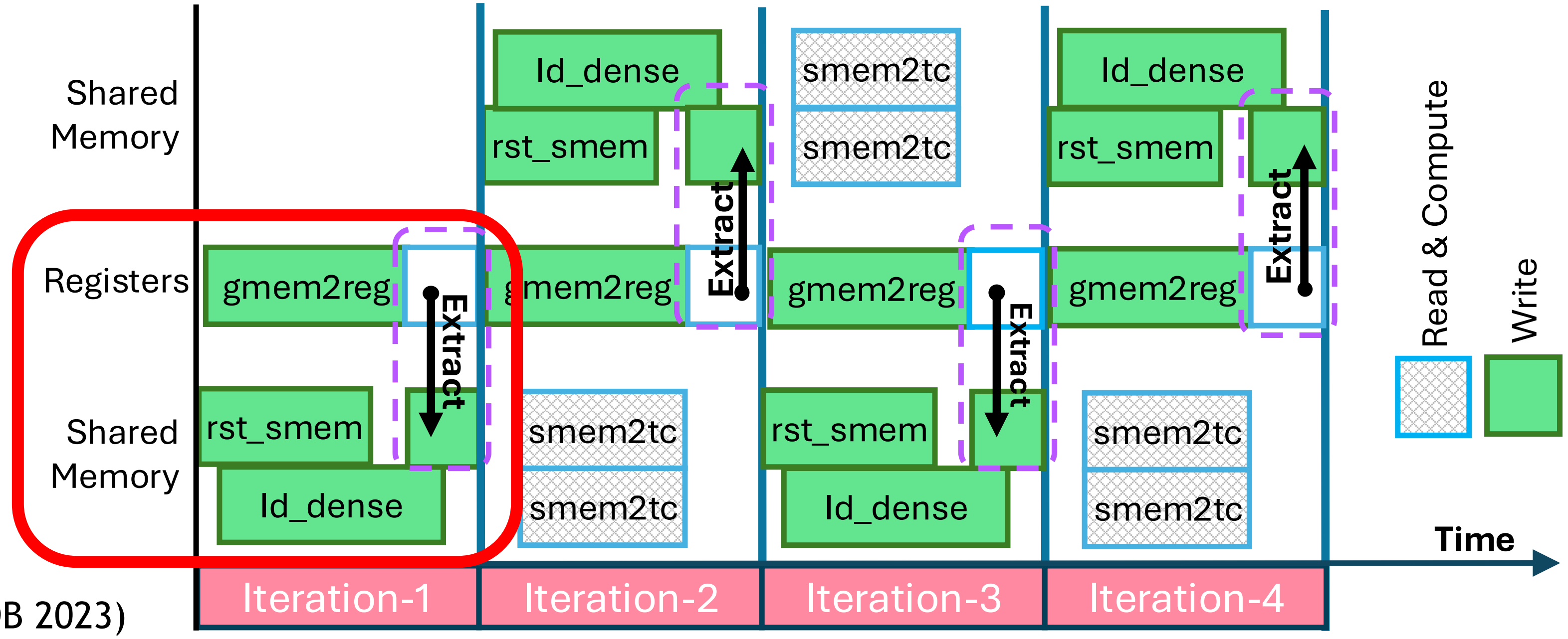


Figure credit to
Flash-LLM (Xia et al. VLDB 2023)

Coruscant GPU Kernel - Load-Compute Pipeline

- Phase 2: Decompressed matrix tiles are computed in Tensor Core.
- Each iteration is an overlapped execution of load and compute.

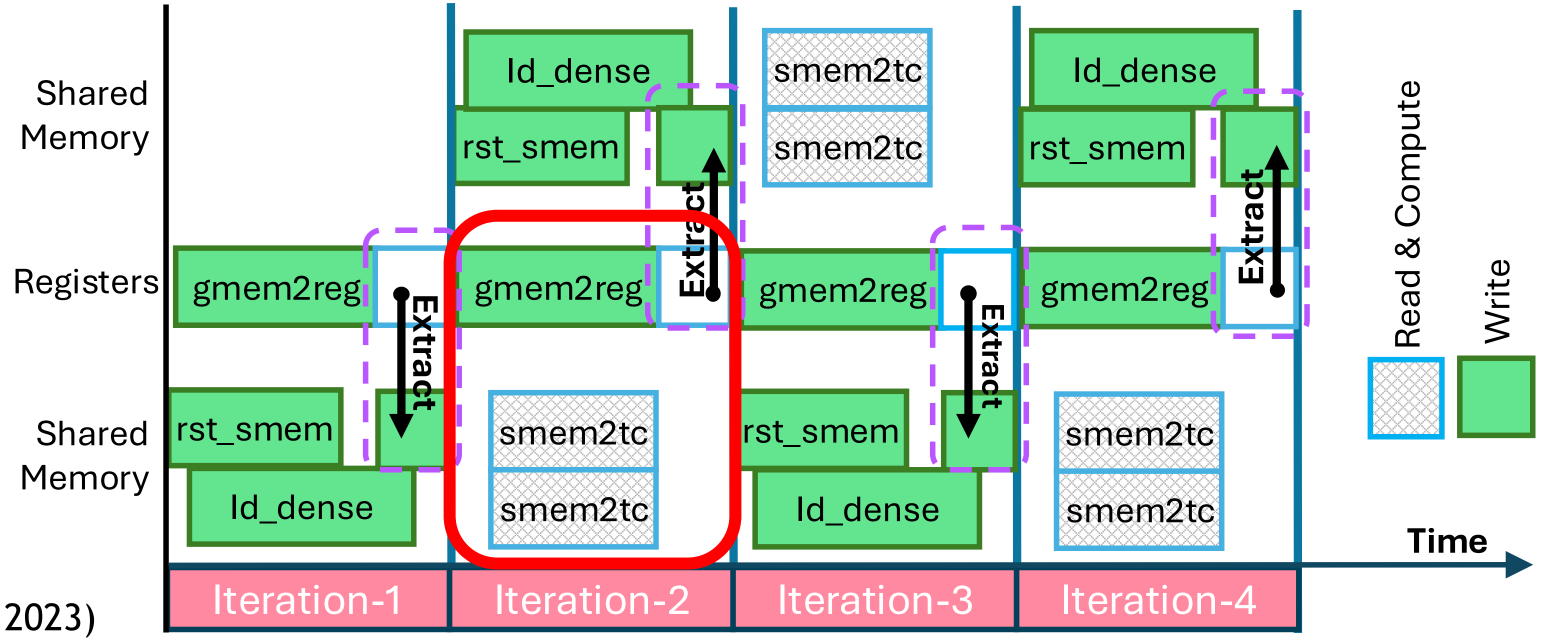


Figure credit to
Flash-LLM (Xia et al. VLDB 2023)

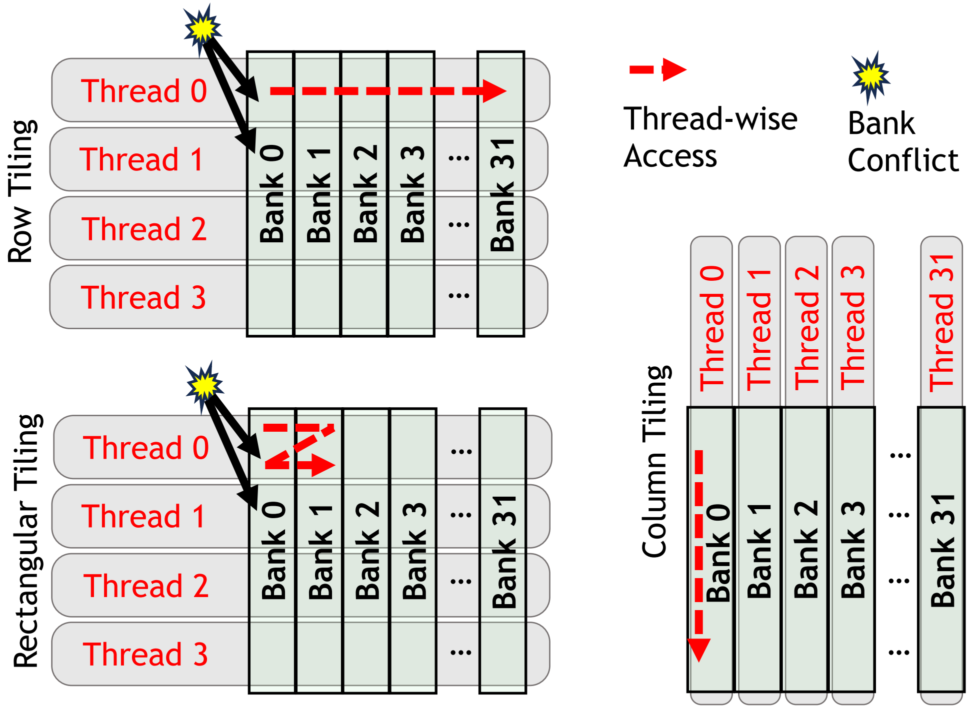
Coruscant GPU Kernel – Decompression Logic

- Decompression loop using count-leading-zero instructions to find set bits quickly.
- Column-wise tiling of Coruscant sparse format prevents shared memory bank conflict.

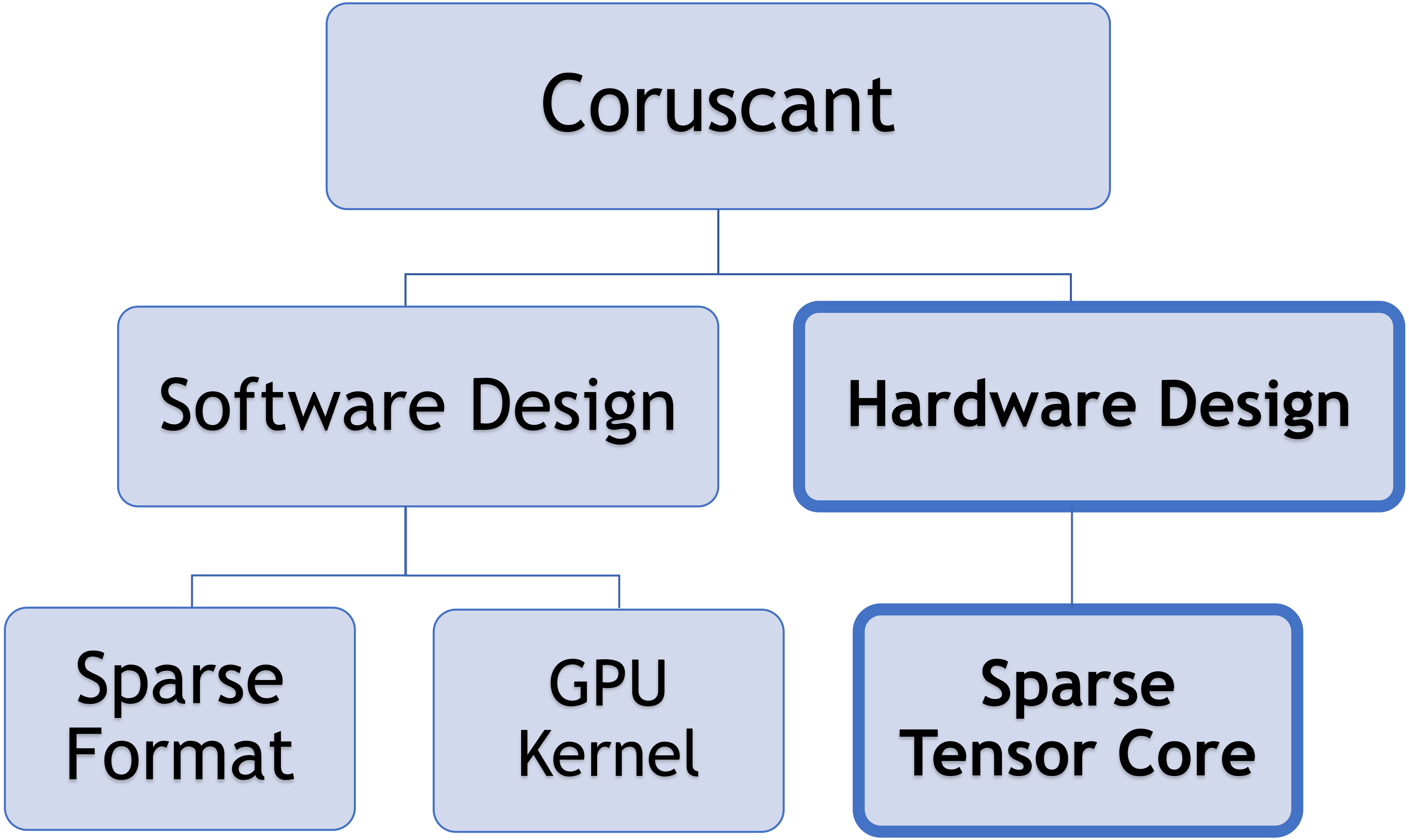
Algorithm 1 Bitmap decompression algorithm of a tile

$\boxed{\textbf{for } i \leftarrow 0 \textbf{ to } 63 \textbf{ do}}$
    if $i = nnz\_tile$ **then**
        break
    end if
    $pos1 \leftarrow \textbf{clz}(bmp)$
    $mask \leftarrow (1 \ll (63 - pos1))$
    $bmp \leftarrow bmp \ \& \ \sim mask$
    $output\_idx \leftarrow tile\_idx + (pos1 \ll 6)$
    $SmemPTR[output\_idx] \leftarrow Reg\_NZs[i]$
end for

Variable loop iteration (i.e. number of NZs) cause register spilling.



Comparing Tiling Schemes on SMEM Write


Coruscant
Software Design
Hardware Design
Sparse Format
GPU Kernel
Sparse Tensor Core

Coruscant Sparse Tensor Core - Motivation

- Software decompression takes up to 36% of kernel execution time.
- In each load-compute pipeline iteration, decompression adds latency.

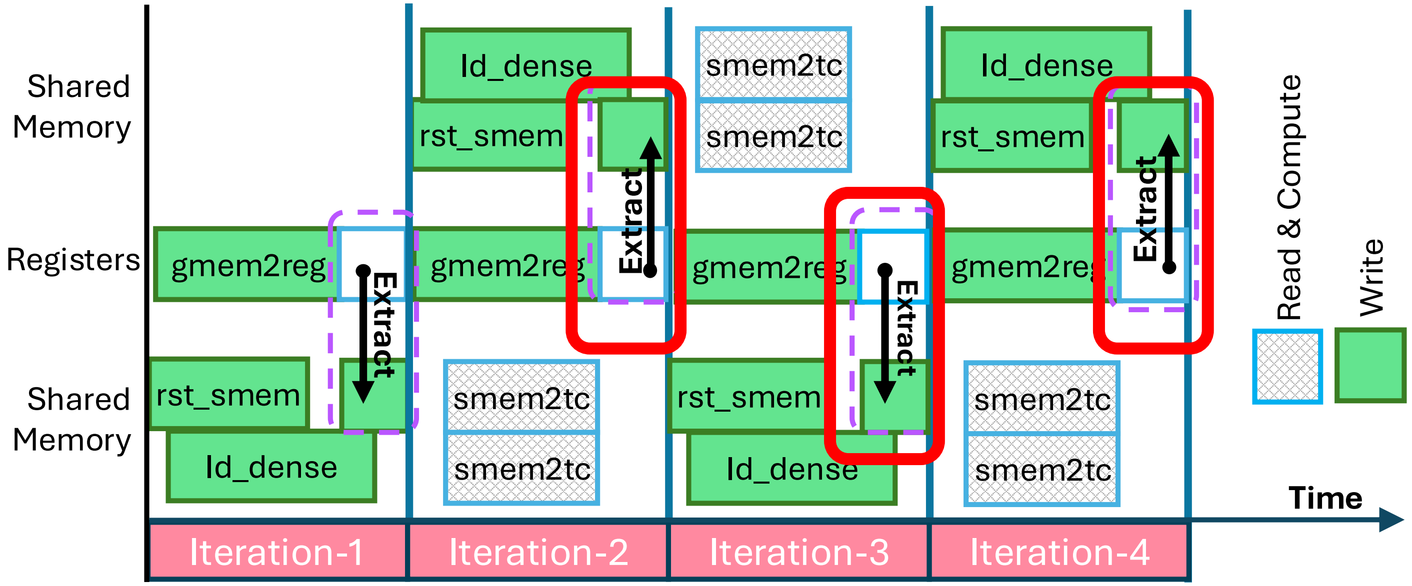
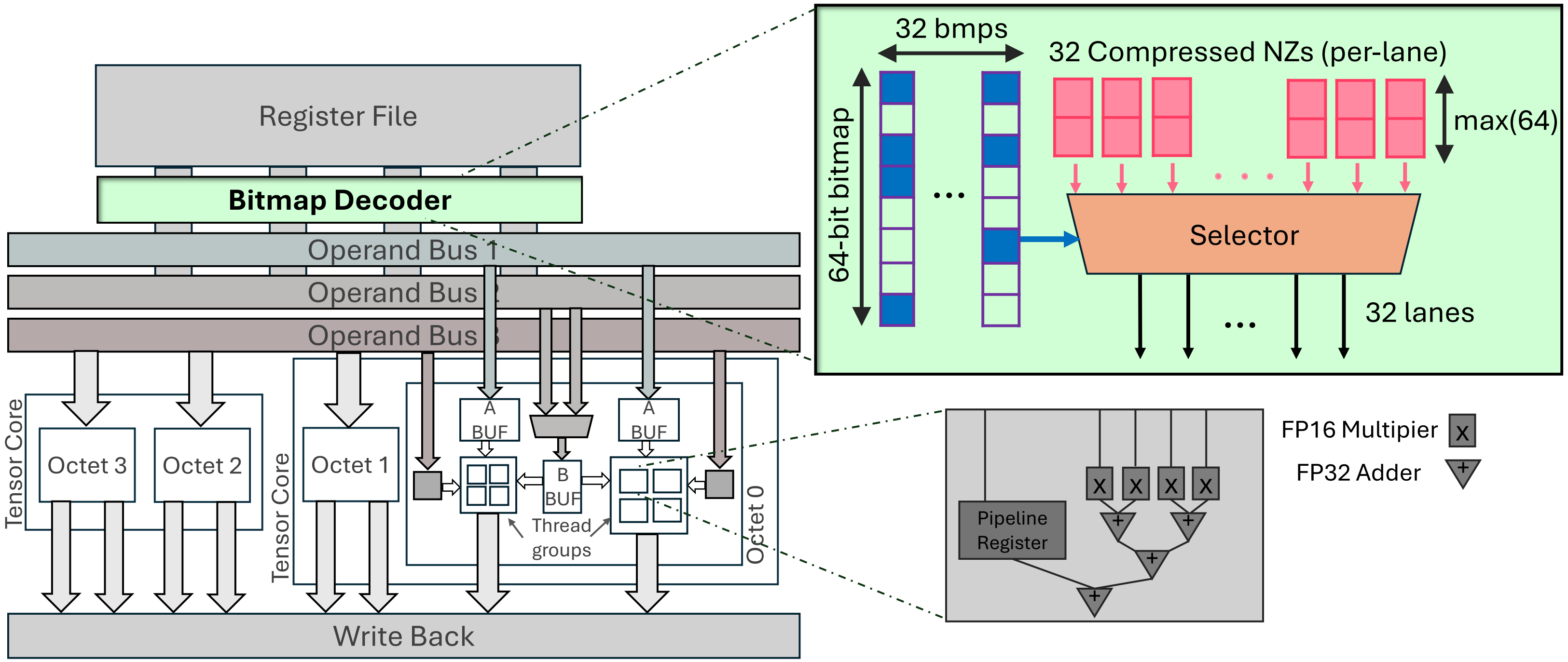
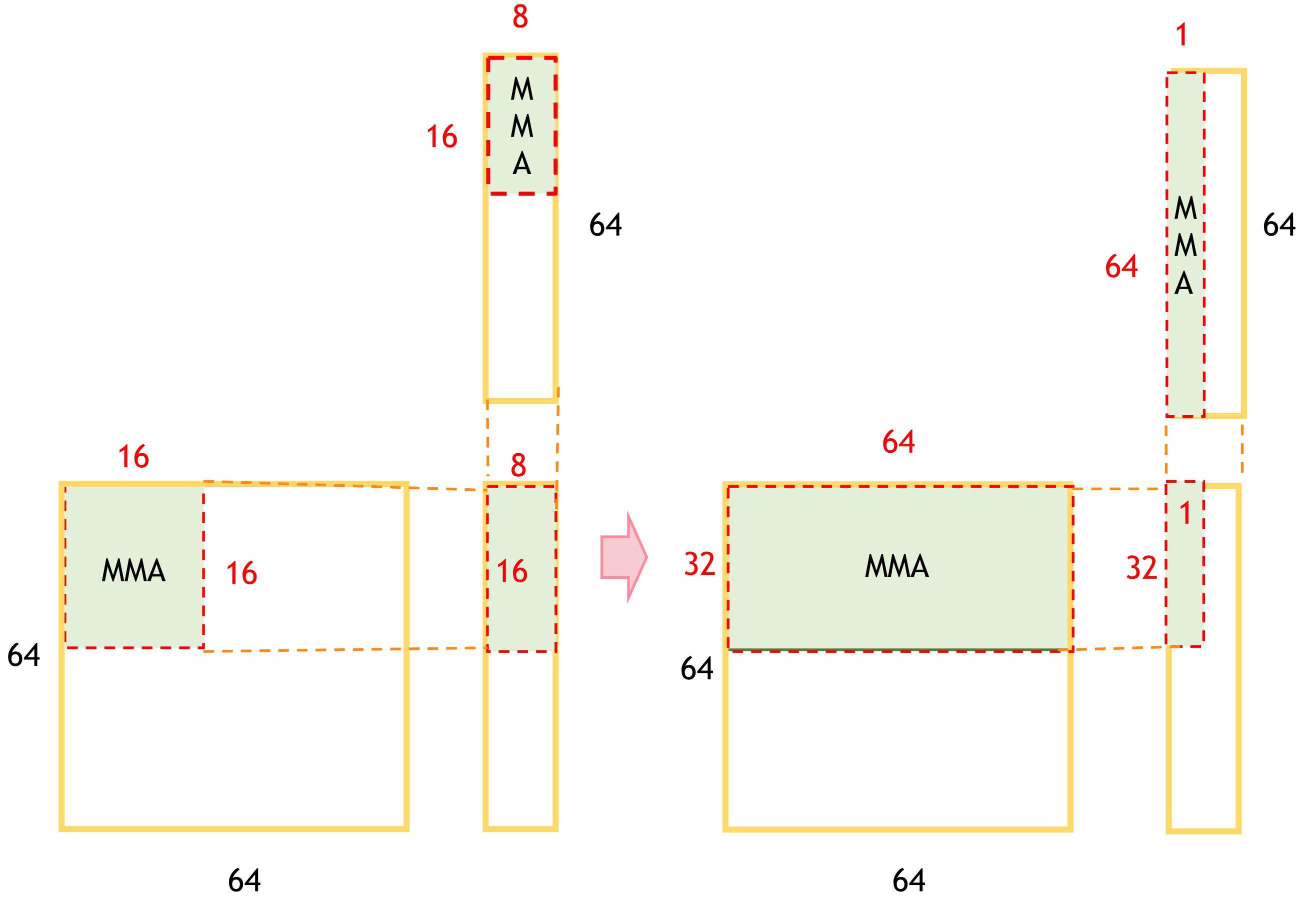


Figure credit to
Flash-LLM (Xia et al. VLDB 2023)

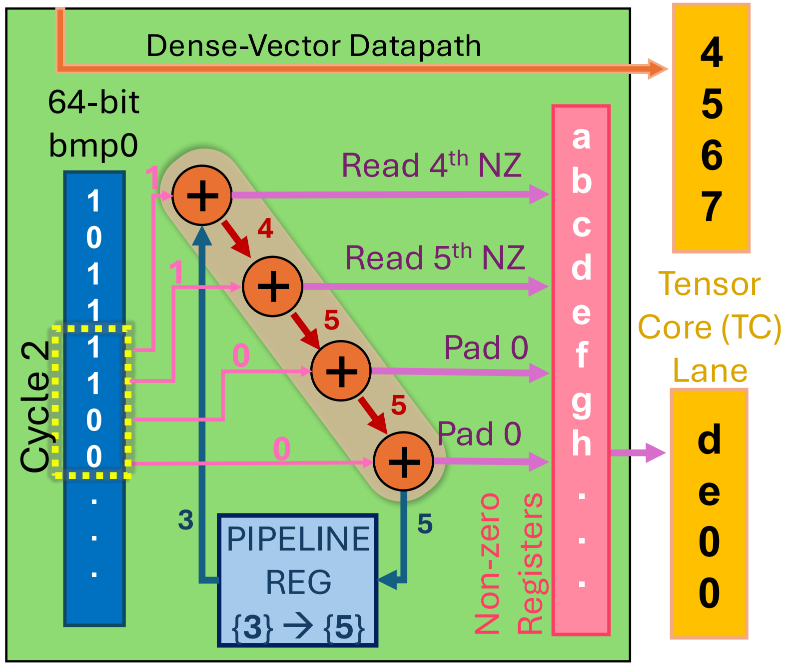
Coruscant Sparse Tensor Core

- Lightweight Bitmap Decoder to perform hardware decompression.
- Decodes the thread-local bitmap to stream non-zeros to compute lanes.





Warp-Level MMA Formulation

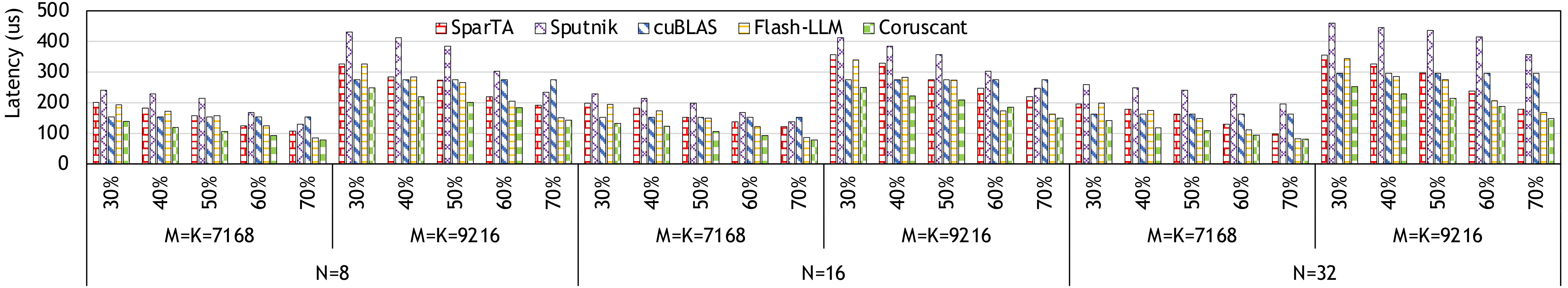


Per-thread Bitmap Decoder

Evaluation Methodology

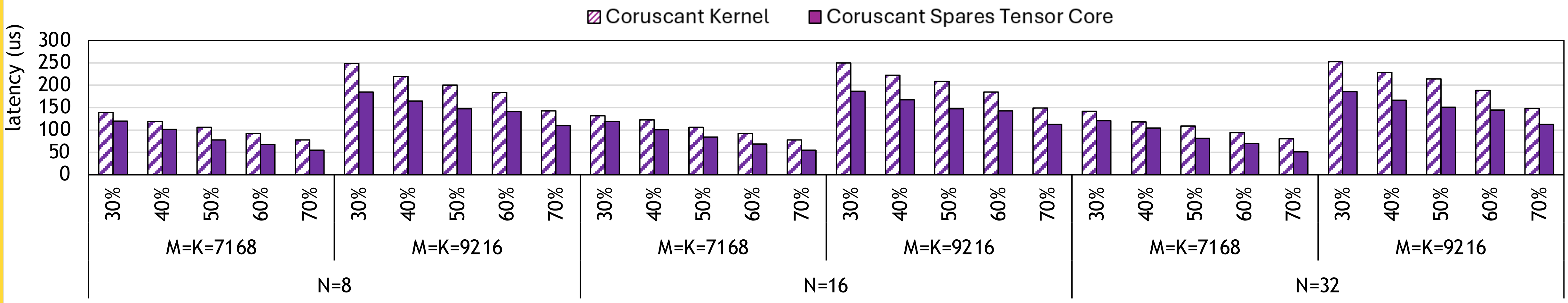
- **System**: NVIDIA RTX 6000 ADA GPU
- **STC Simulation Methodology**: Coruscant STC modeled as a kernel without decompression, but equal number of MMA instructions.
- **Key Metrics**:
  - **Efficiency**: SpMM/end-to-end latency, GMEM consumption, tensor core utilization
  - **LLM Quality**: Perplexity on Wiki-Text, TriviaQA accuracy from Longbench
- **Workloads**:
  - SpMM evaluation: Weight projection matrices of OPT 13B and 66B
  - End-to-end inference: Llama-2 7B and 13B

Evaluation: Coruscant Kernel



- Coruscant Kernel outperforms dense and sparse SOTAs across target sparsity.
- Outperforms dense cuBLAS with 1.09x (at 30%) to 2.00x (at 70%) speedup
- Outperforms sparse FlashLLM with 1.05x (at 70%) to 1.48x (at 30%) speedup

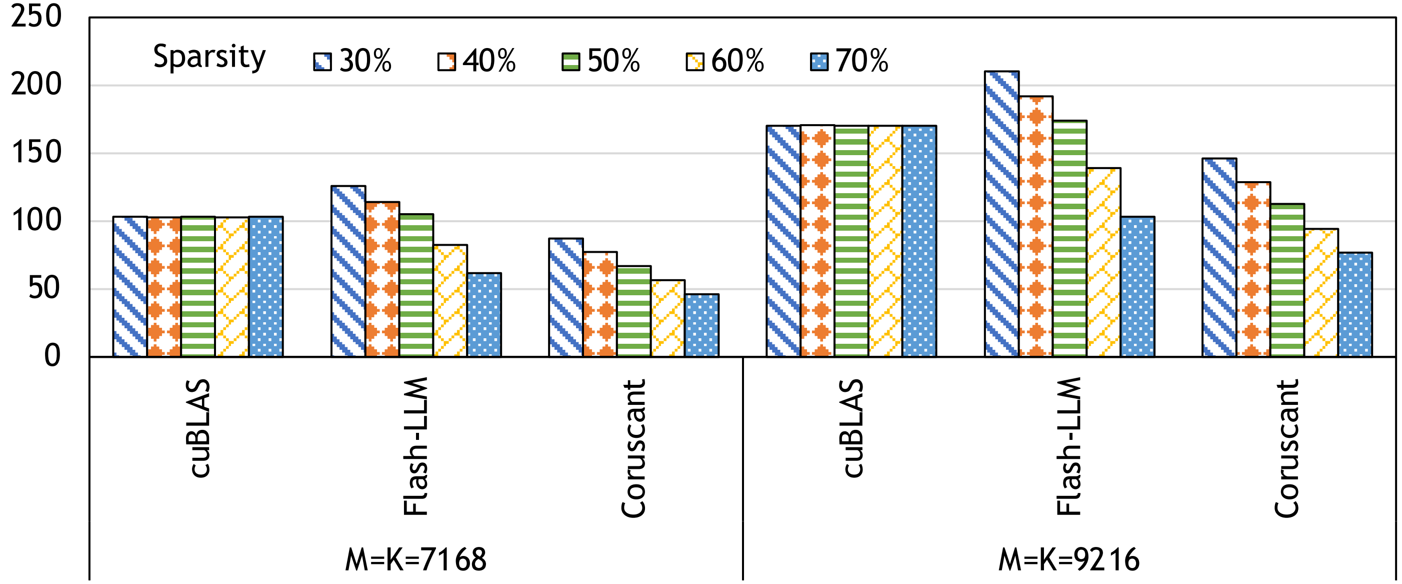
Evaluation: Coruscant Sparse Tensor Core



- Hardware decompression achieves average 1.3x speedup over Coruscant Kernel.
- Removing decompression latency improves Tensor Core utilization.
- Reducing shared memory read/write improves shared memory stalls.

Coruscant in Roofline Analysis

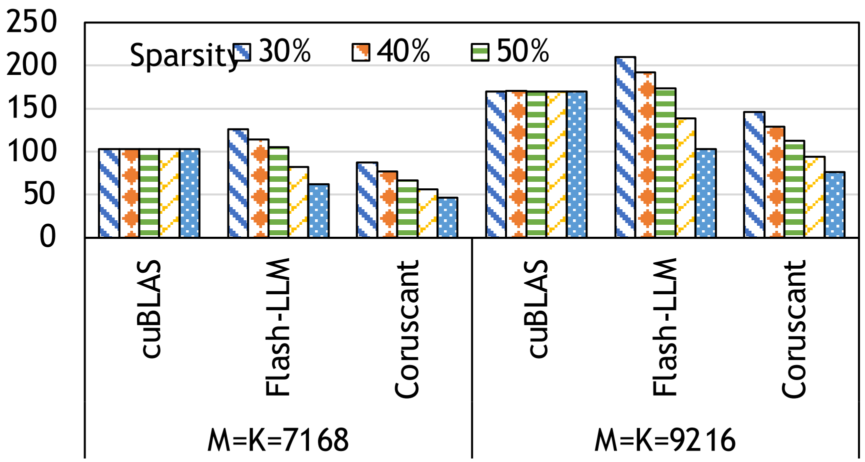
- Speedup is primarily attributed to bitmap compression reducing data transfer.



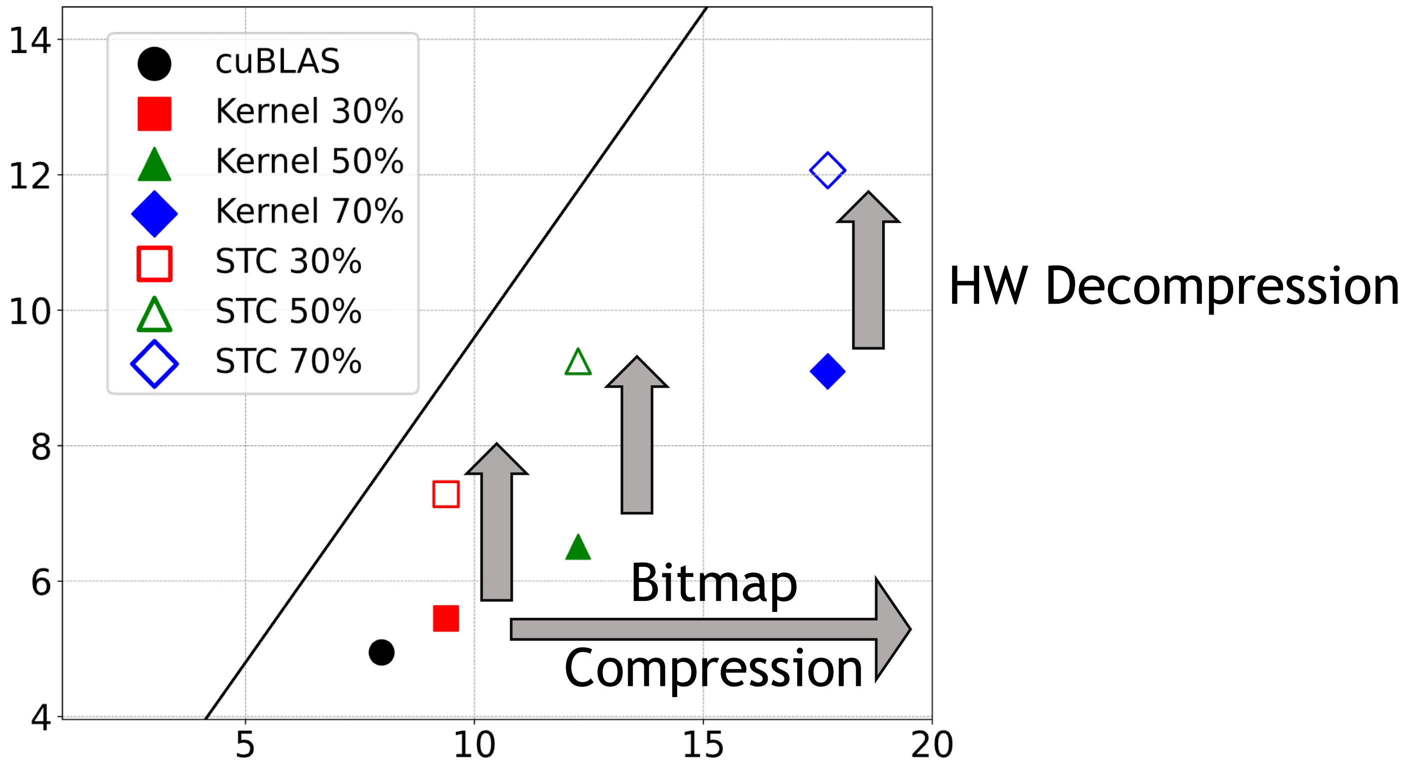
Kernel Memory Footprint (MB)

Coruscant in Roofline Analysis

- Speedup is primarily attributed to bitmap compression reducing data transfer.
- In roofline, hardware decompression represents an upright movement.

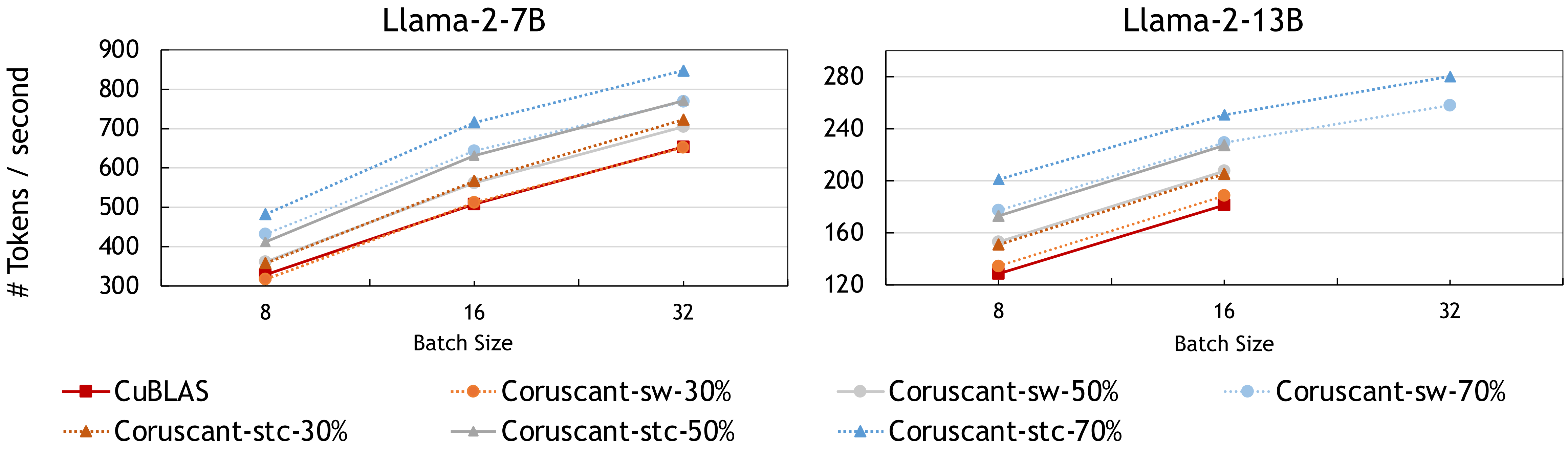


Kernel Memory Footprint (MB)



Evaluation: End-to-End Inference

- Llama-2-7B: Up to 26% increase in token generation throughput.
- Llama-2-13B: Up to 40% increase in token generation throughput.
- Reduces global memory footprint to support larger batch size.



d-Matrix
DEPARTMENT OF COMPUTER SCIENCE

Additional Evaluations

- Please refer to our paper for evaluation and analysis on …
  - Latency, power, and area comparison with previous STC designs
  - Sparsity exploitation characterization: compute-skipping versus compression
    - Comparison between Coruscant and 2:4 semi-structured sparsity kernel
    - Comparison between Coruscant and N:M semi-structured accelerators
  - Performance on different workloads (Summarization, Conversational, Reasoning)
  - Impact on weight transfer over PCIe.
  - and more!

Conclusion

- Coruscant is a **software-hardware co-design** that keeps unstructured sparse weight matrices compressed from global memory to tensor core execution.

- **Contributions:**
  - ✓A **bitmap-based sparse format** for compressing unstructured sparsity
  - ✓A **GPU kernel** to perform *load-as-compressed, compute-as-dense*
  - ✓A **sparse tensor core** to eliminate decompression latency

- Demonstrates that **unstructured sparsity can be practical and efficient** on modern GPUs with minimal hardware cost.

Thank You

We are happy to take

Questions and Comments

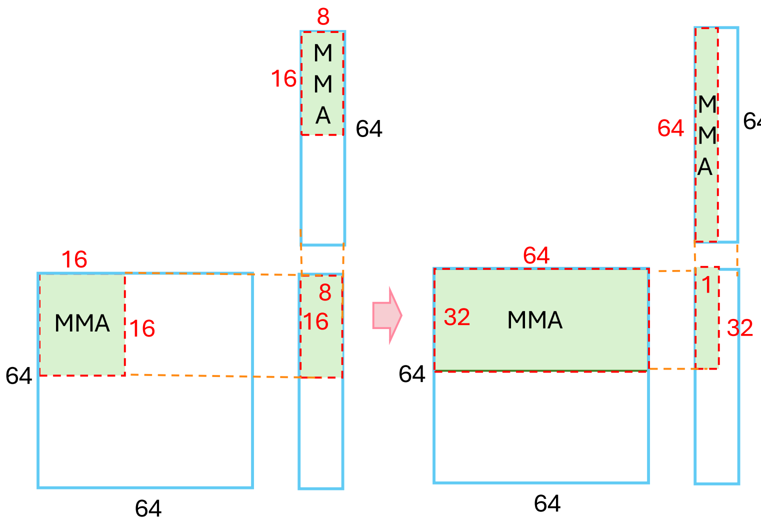
dhjoo98@umd.edu

This work is supported by
National Science Foundation (NSF).

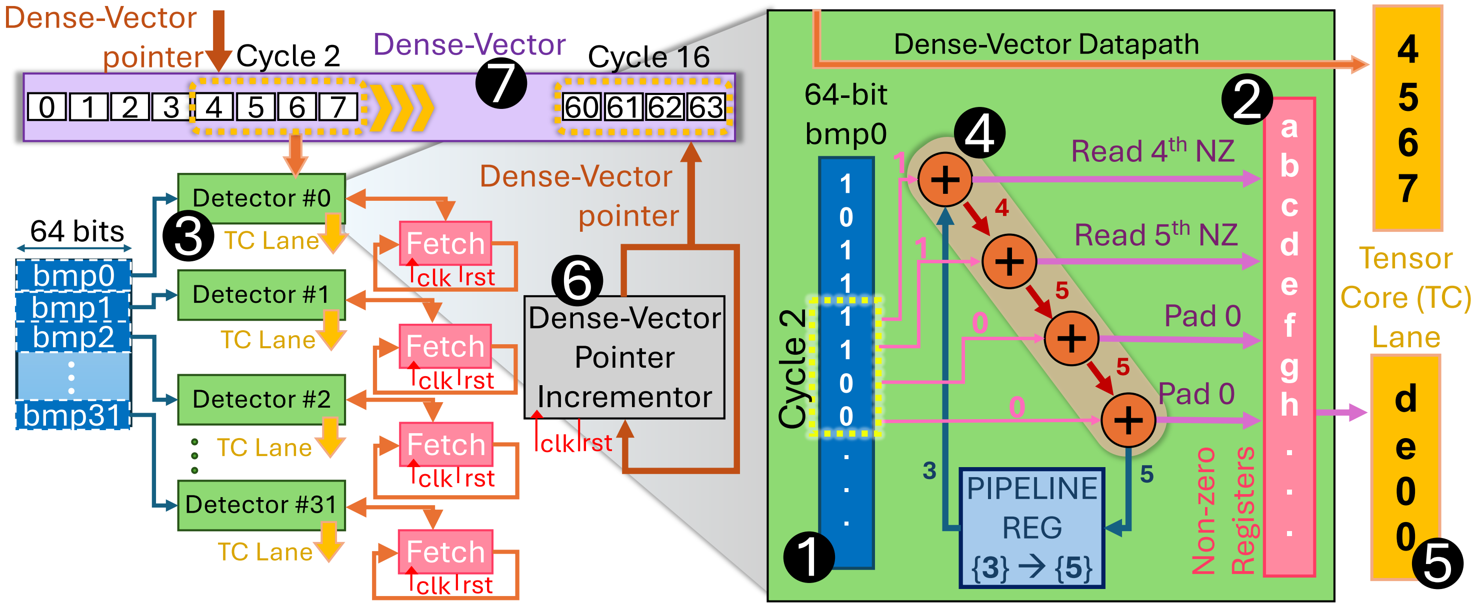
Backup Slides

Coruscant Sparse Tensor Core - Bitmap Decoder

- Warp-level MMA formulation is altered to stream compressed non-zero from register to TC.
- Consecutive 4-bits are decoded every cycle to stream a decompressed 4-element vector.



(a) Warp-Level MMA Formulation



Analysis: Compute Skipping vs Reducing Data Movement

- Two ways to leverage sparsity:
  - Compute Skipping accelerates computation.
  - Sparse matrix compression reduces memory footprint.

- Previous STC designs for unstructured sparsity prioritized Compute Skipping.
- Architecture for semi-structured sparsity shows strength in both.
- Coruscant prioritizes maximal compression, targeting memory-bound SpMM.

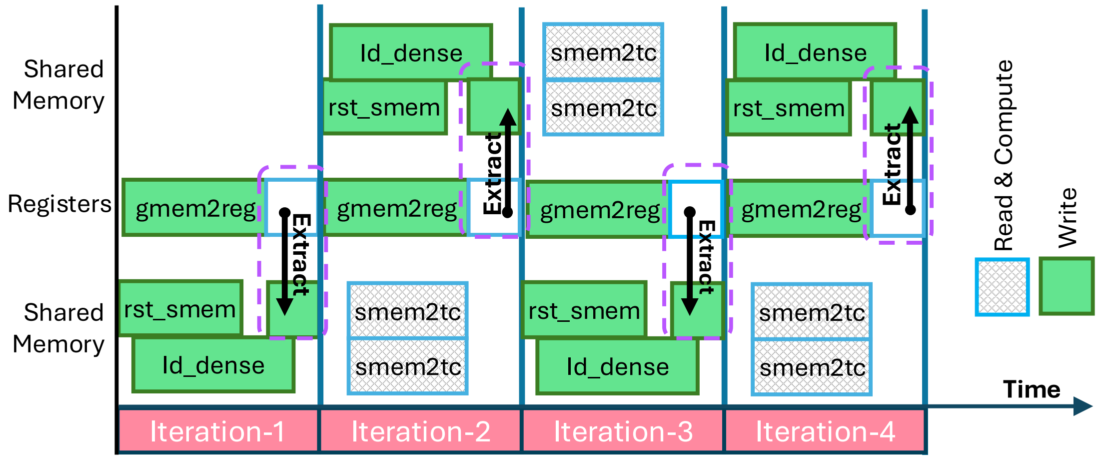
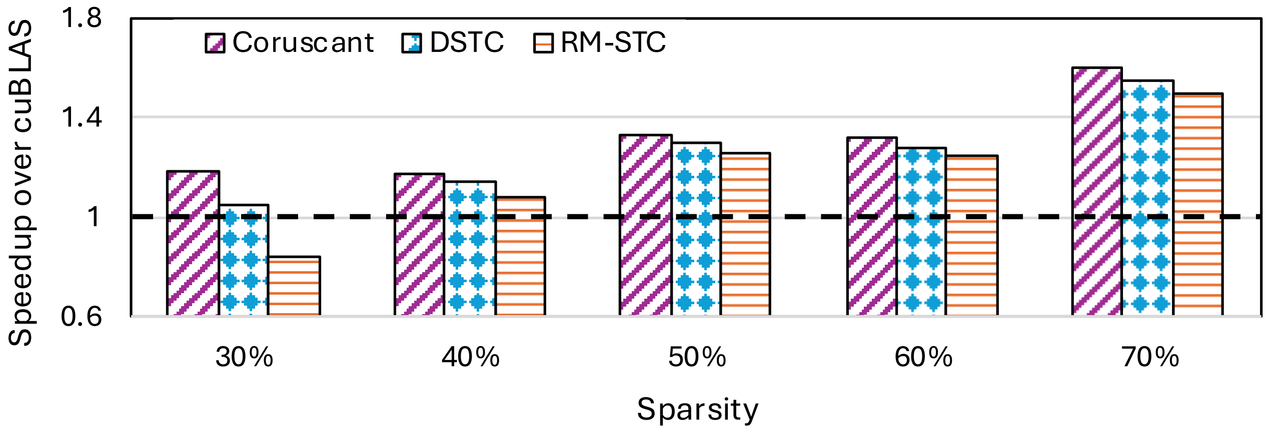


Figure credit to
Flash-LLM (Xia et al. VLDB 2023)

Analysis: In-depth Comparison with Previous STCs

- Previous STC designs (DSTC, RM-STC) for unstructured sparsity seek to skip computation.
- Smaller tile size leads to 6%/12% non-zero pointer size, deteriorating compression ratio.
- This directly impacts performance on memory-bound SpMM.
- Compute-skipping increase the complexity of accumulation logic, increasing HW overhead.



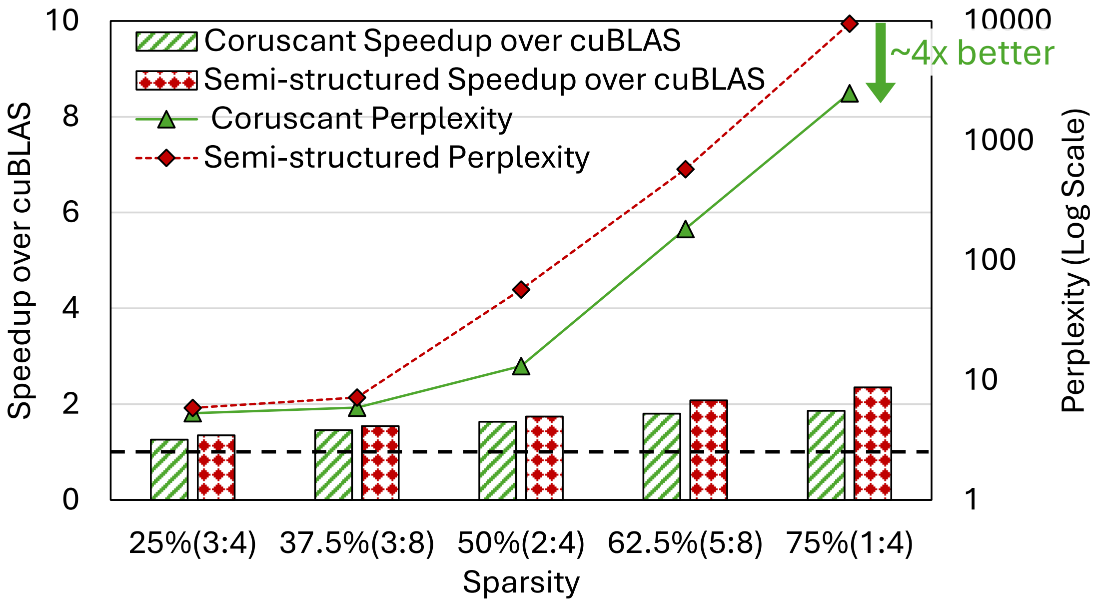
	Added Area	Percentage to GPU	Power
Volta (V100)	0.51 mm^2	0.006%	26.24 mW
Ampere (A100)	0.68 mm^2	0.008%	35.43 mW
Hopper (H100)	1.44 mm^2	0.018%	74.79 mW

Analysis: Comparison with Semi-Structured Sparsity

- Efficiency – model accuracy trade-off applies to various N:M patterns for different sparsity.
- Semi-structured sparsity can compress better, but model accuracy degrades significantly.
- Coruscant frees future pruning works from sparsity pattern constraints

Table 4: TriviaQA score of semi-structured and unstructured sparsity at various sparsity levels.

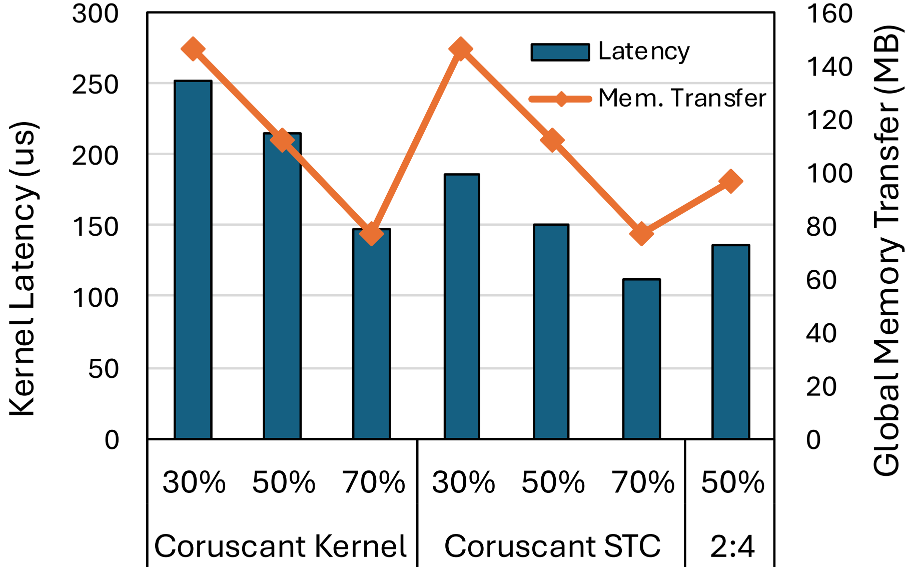
	25% (3:4)	37.5% (3:8)	50% (2:4)	62.5 % (5:8)	75% (1:4)
Semi-Structured	84.22	75.70	6.09	0.06	0.00
Unstructured	**88.40**	**84.43**	**42.81**	**2.21**	**0.43**

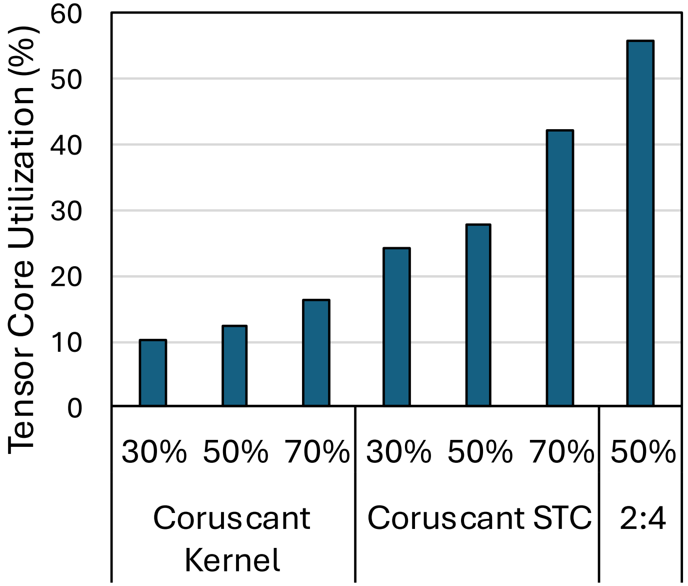


Industry	--	--	NV-STC	--	--
Academia	VEGETA	S2TA	S2TA VEGETA	S2TA	VEGETA

Analysis: In-depth Comparison with 2:4 cuSPARSELt

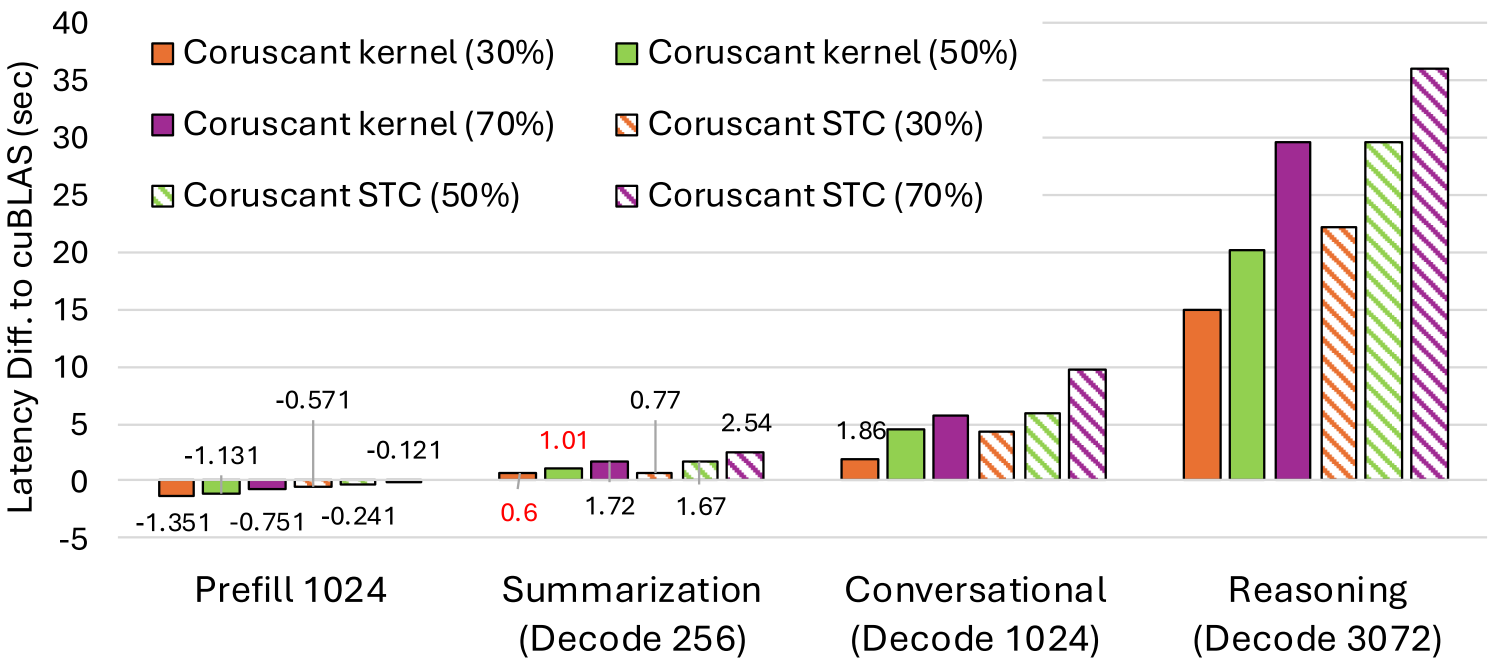
- cuSPARSELt leverages the hardware decompression support from NVIDIA STC.
- Compared to Coruscant Kernel-only, cuSPARSELT dominates performance and TC utilization.
- Coruscant STC shows comparable performance to 2:4 semi-structured sparsity, while being flexible to arbitrary sparsity.





Evaluation: Generality and Workload Types

- Due to low occupancy, Coruscant underperforms for compute-bound prefill.
- Across representative input-output token ratio, slow prefill is amortized.



d-Matrix

DEPARTMENT OF COMPUTER SCIENCE