Cheap Plans, Costly Trades: Host–Device Tradeoffs for SpGEMM on AiM

Calvin Jialin Zhang [1], Giulia Guidi (Advisor) [1]
[1] Cornell University

CornellBowers Computer Science

ALPS Lab

Problem Statement

SpGEMM (C = A·A) is memory-bandwidth-bound. **What does it take to run SpGEMM on SK hynix's Accelerator-in-Memory (AiM), and does host planning that shrinks the device's work shrink the total time?**

Main Contribution

- Outer-product SpGEMM on an AiM simulator.
- Two costs it pays: routing and a write floor.
- A single packing pass cuts device cycles 7–122×; the best design reaches 19–1027× over naive.

AiM Hardware

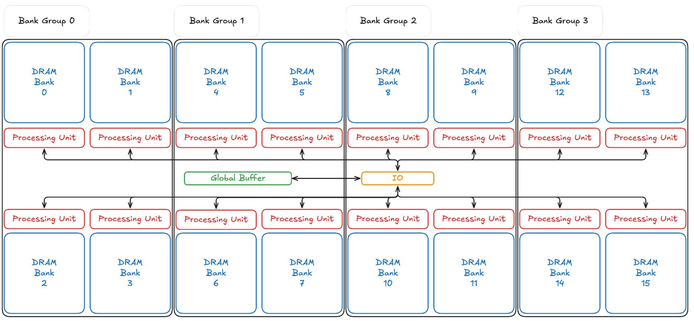


Fig. 1: One AiM channel

$C = AA$ **SpGEMM Perspectives**

Inner Product ❌

MAC sums all 16 lanes into one accumulator, so a cell's lanes must all feed the same sum, aligned by k and zero-padded.

$$c_{ij} = \sum_{k=1}^{n} a_{ik}\, a_{kj} = A_{i,:} \cdot A_{:,j}$$

Outer Product ✅

EWMUL keeps all 16 products separate, and the host chooses which cell each nonzero occupies, so up to 16 nonzeros share one 32 B cell.

$$C = \sum_{k=1}^{n} \mathbf{a}_k\, \mathbf{b}_k^{\top}, \quad \mathbf{a}_k = A_{:,k}, \quad \mathbf{b}_k^{\top} = A_{k,:}$$

The Core SpGEMM Mapping on AiM

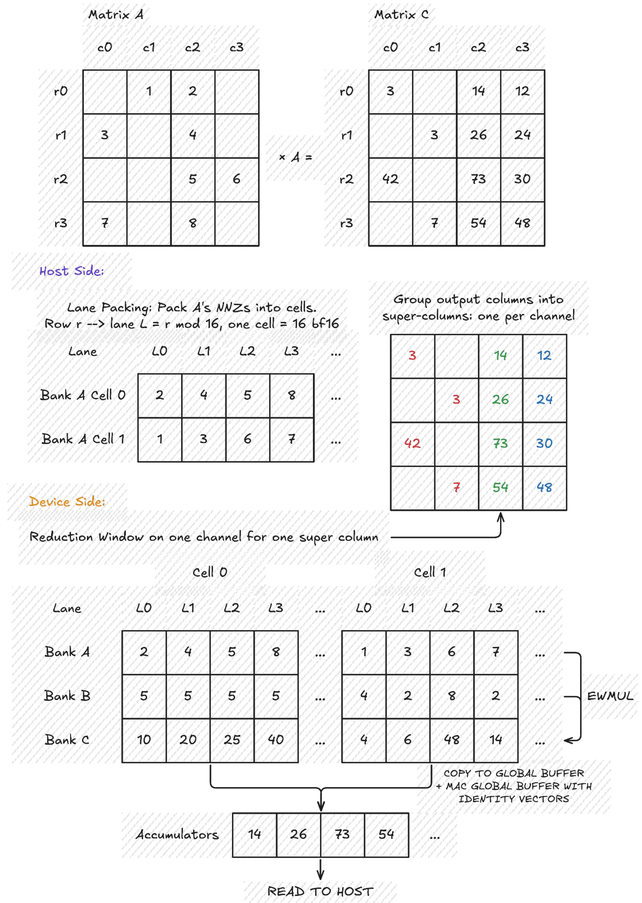


Fig. 2: Core routing and optimizations for SpGEMM on AiM

Matrix	Domain	Rows	Device cycles (32 channels) S1 naive	S2 pack	S3 plan	S4 search	S1 → S2	S1 → S4
GD99_b	graph	64	11.8K	1,597	782	610	7×	19×
GD96_1k	graph	1,096	556K	22.7K	4,525	3,611	25×	154×
can_256	structural	256	816K	33.4K	14.8K	13.3K	24×	61×
G32	random	2,000	7.23M	59.0K	7,042	7,040	122×	1027×
SciMet	citation	3,084	6.29M	480K	101K	92.0K	13×	68×
G29	random	2,000	26.8M	1.79M	485K	483K	15×	55×
bitcoin	social	3,783	67.4M	4.88M	2.29M	2.10M	14×	32×
smallworld	small-world	100K	1.26B	11.7M	3.07M	3.04M	108×	416×
roadNet-PA	road	1.09M	2.58B	53.6M	6.66M	6.49M	48×	398×
belgium_osm	road	1.44M	1.71B	44.2M	5.22M	5.22M	39×	327×
webbase-1M	web	1.00M	4.03B	205M	280M	261M	20×	15×

Fig. 3: Device cycles for all four designs: 32 channels, different host side plan

Results

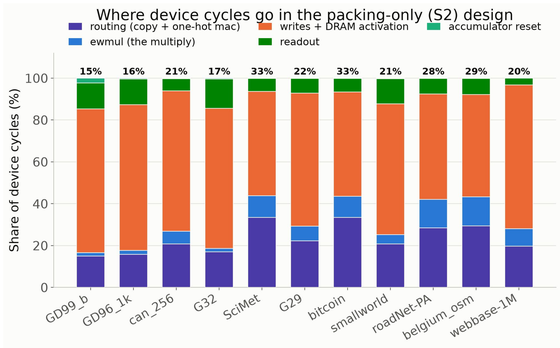


Fig. 4: Where S2's device cycles go

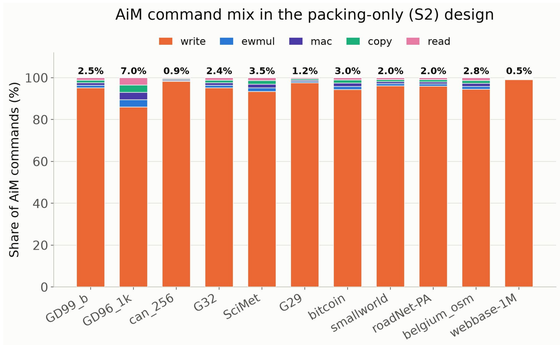


Fig. 5: AiM command mix in S2

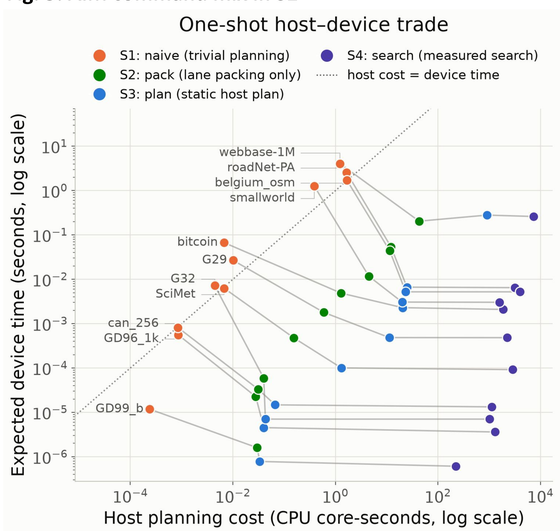


Fig. 6: Host planning cost vs. device time, one line per matrix

Conclusions

- **Outer product fits:** EWMUL needs no k-alignment, so the host packs nonzeros into cells.
- Packing is the win: **7–122×.** Plan adds 2.0–8.5× (except webbase-1M); search at most 1.28×.
- Writes are the floor; **routing is the removable cost** (15–33% of cycles in the pack design).
- **No plan pays off on one product**.

Future Work

- **Fused routing.** Let EWMUL add each product straight into its lane's accumulator, collapsing EWMUL + copy + one-hot MAC into one.
- **General C = A·B.** Extend the mapping to rectangular matrices.
- **Pattern reuse.** Does a plan built once pay off when only the values change, as in AMG setup?

References

1. Davis & Hu. The university of Florida sparse matrix collection. ACM TOMS 38(1), 2011.
2. SK hynix MSPD. Accelerator-in-Memory: SK hynix's GDDR6-based PIM device. Whitepaper, 2022.
3. Kwon et al. A 1ynm 1.25 V 8 Gb 16 Gb/s/pin GDDR6-based accelerator-in-memory. IEEE JSSC 58(1), 2023.
4. Luo et al. Ramulator 2.0: A Modern, Modular, and Extensible DRAM Simulator. IEEE CAL 23(1), 2024.

Acknowledgments

We thank Irene Simó Muñoz for her valuable feedback on this project and her expertise in sparse algorithms on AiM.