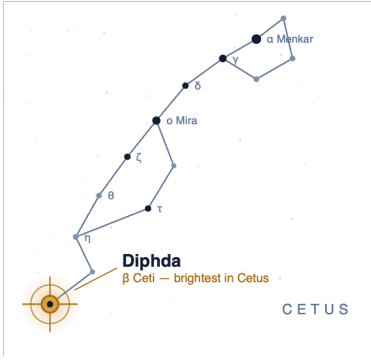




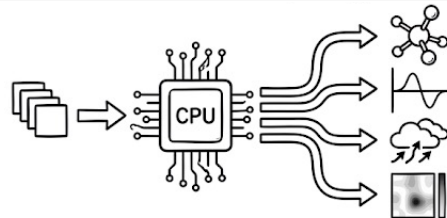
Diphda: Demystifying Processing- in-HBM for SpGEMM across Diverse Data Mapping

Helya Hosseini^{*}, Sanjali Yadav^{*}, Christina Giannoula[†], Bahar Asgari^{*}

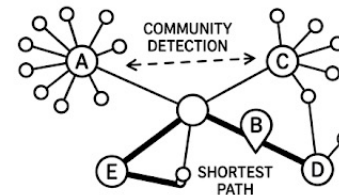
^{*} University of Maryland, College Park

[†] Max Planck Institute for Software Systems

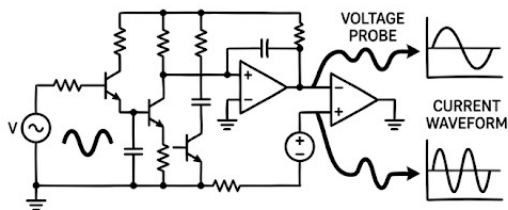
Sparse computation is everywhere



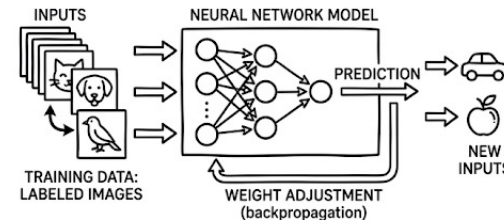
**Scientific
computing**



**Graph
analytics**



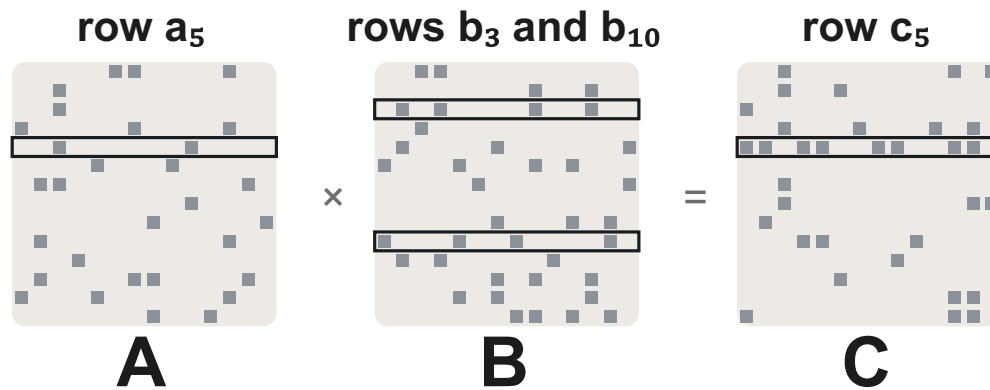
**Circuit
simulation**



**Machine
learning**



SpGEMM: Where both operands are sparse



The row-wise rule

$$c_5 = \sum A(5,k) \cdot b_k \quad \text{for every } k \text{ where } A(5,k) \neq 0$$

Irregular access

**Low arithmetic
intensity**

Skewed work

HBM-PIM: A PIM unit reads and writes its own bank only.

Compute is bank local

DRAM die · banks + compute units

Re...

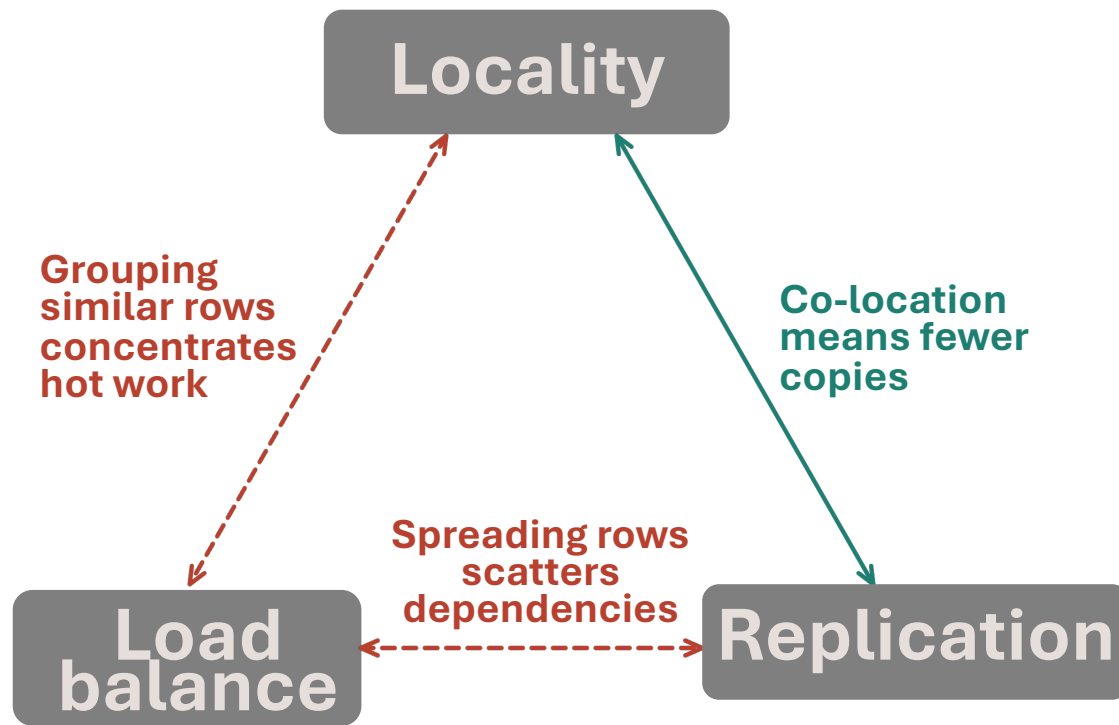
Operands must be co-located with the computation,
so **data placement** becomes crucial.

Data is replicated

Logic layer

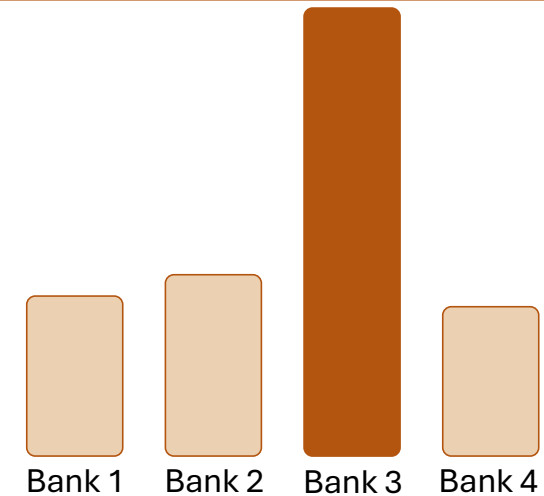
Bank needs b5

Three trade-offs in data placement



Why load balance matters?

latency = slowest bank



Seven answers to the same questions

Baselines: balance the work, never inspect the data

	What it looks at?	Best suited to?
M0a: Consecutive	Position only: A is cut into equal contiguous blocks, one per bank. Zero cost.	Matrices whose related rows are already neighbors. i.e., banded or diagonal
M0b: Round-robin	Position, cyclically: row i goes to bank i modulo the bank count.	Matrices whose heavy rows arrive in contiguous runs that a block cut would not break up.
M0c: NNZ-greedy	Work only: each row goes to whichever bank currently holds the least work.	Workloads where the arithmetic, not the fetch, is what costs, dense enough that compute dominates.

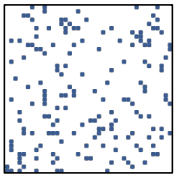
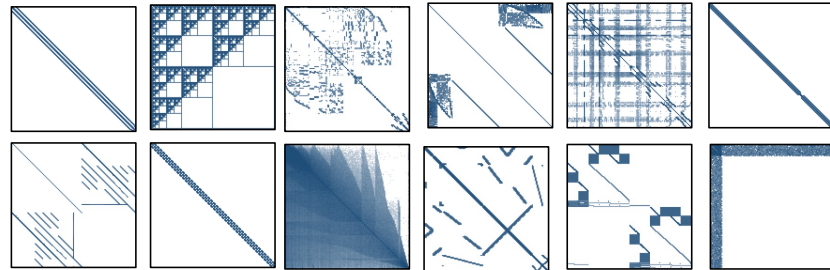
Dependency-aware: inspect rows of B needed by rows of A

	What it looks at?	Best suited to?
M1: Rarest anchor	How often each B-row is wanted. B is placed first; each A-row is anchored on its rarest dependency.	Matrices with a sharp split between a few very popular B-rows and a long tail of rare ones.
M2: Similarity grouping	Set similarity: each row's dependency set is hashed to a signature, and similar rows are grouped.	Matrices where rows share large, overlapping dependency sets, substantial reuse to find.
M3: Coverage greedy	What is already there: each row goes where most of the B-rows it needs already sit, under an imbalance cap.	The general case. It assumes nothing about structure and builds locality incrementally.
M4: Hypergraph greedy	The whole dependency structure at once: B-rows are hyperedges, and the heaviest are cut last.	Matrices with community or block structure, where a global view finds groups, a local rule would miss.

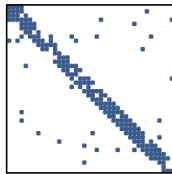
What we measured, and on what

Workloads:

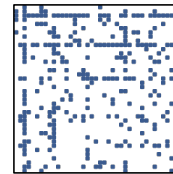
- SuiteSparse
- Synthetic



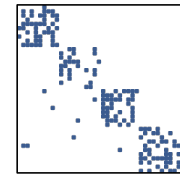
Random



Diagonal



Power-law



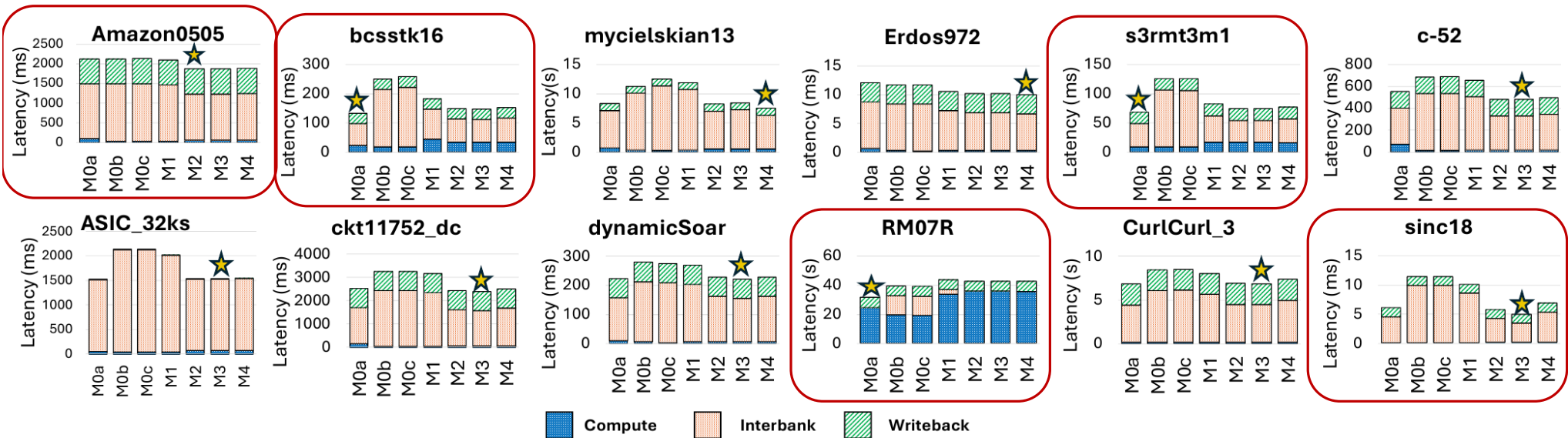
Block structured

Simulator:

- Cycle-accurate simulator based on AttAcc¹ simulator
- Mapping itself runs offline on a host CPU (i9-14900K), timed separately

¹ Park et al., “AttAcc! Unleashing the Power of PIM for Batched Transformer-Based Generative Model Inference,” ASPLOS ’24, 2024.

On real matrices: interbank dominates latency



MS Compute-heavy: M0a wins

W The busiest bank sets the critical path;
r balance wins by 1.36×.

Density shifts which mapping wins

The optimal mapping depends on whether execution is **merge-dominated or compute-dominated**. Dependency-aware mappings (M2–M4) are most effective for sparse, merge-dominated workloads, while balanced mappings become preferable as density increases.

M2/M3 → M0a

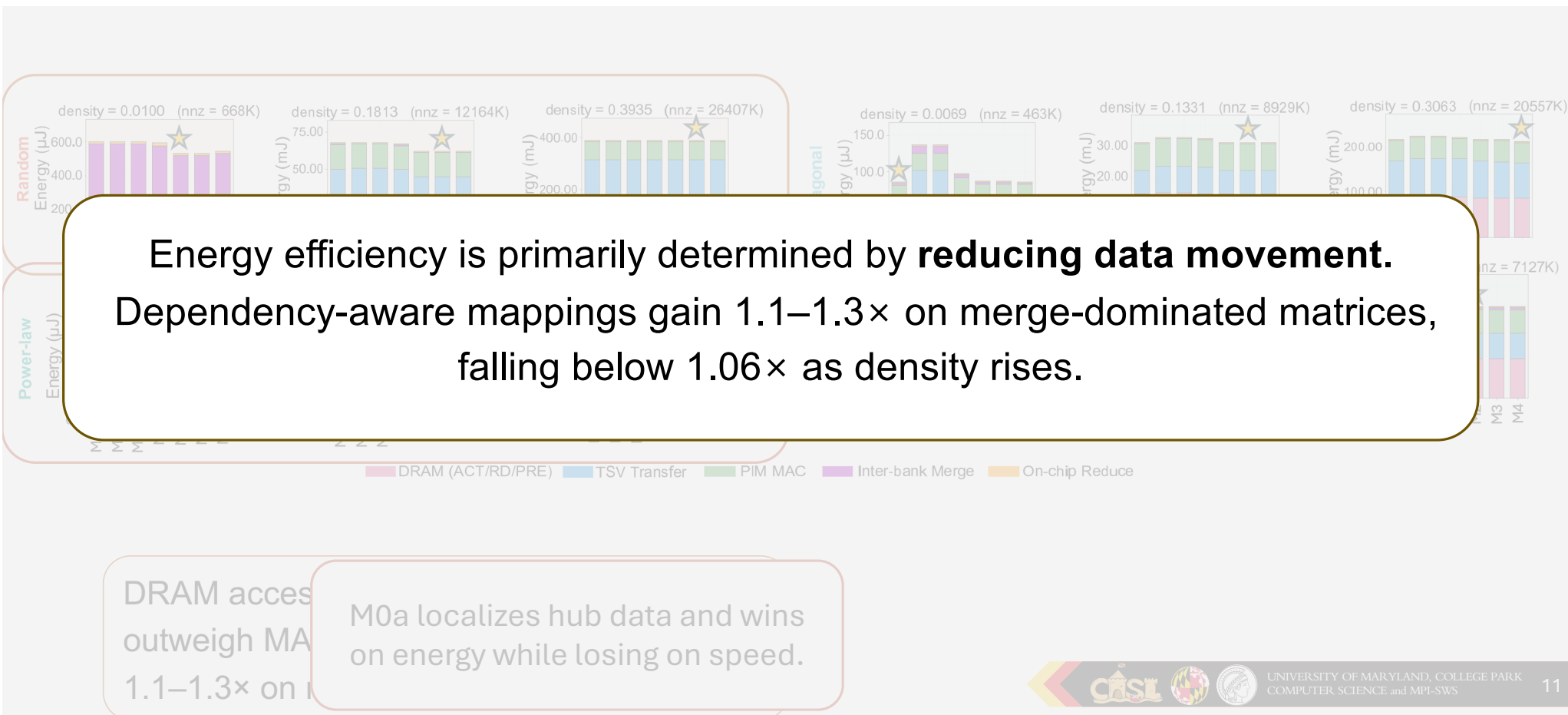
Sparse: merge > 80%, M2/M3 up to 1.16× faster.
Dense: M1 falls 2.49× behind M0a.

M3 / M4

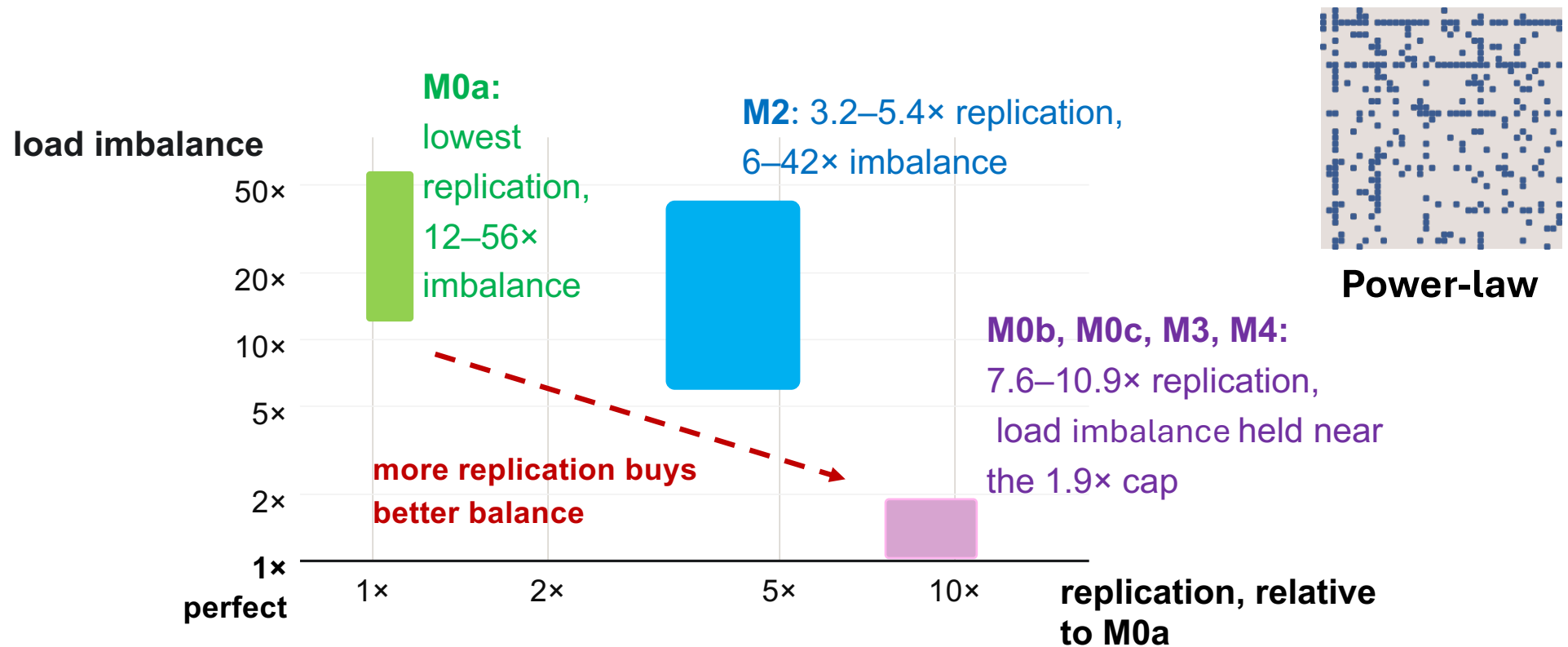
M0a piles hubs into few banks;
M3/M4 up to 3.7× faster.



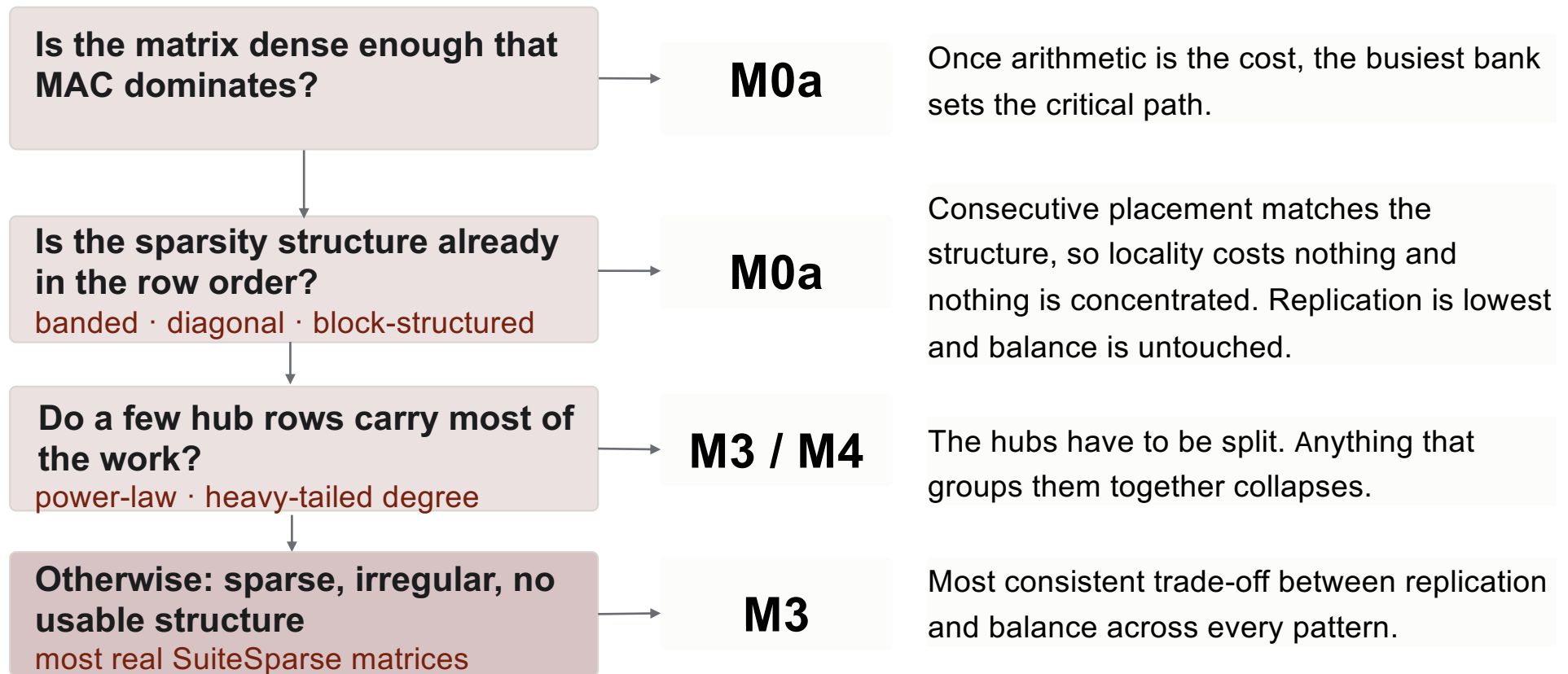
Energy is dominated by data movement



Power-law: locality and balance pull apart



Choosing a mapping in practice



Thank you!

Diphda: Demystifying Processing-in-HBM for SpGEMM across Diverse Data Mapping

Helya Hosseini*, Sanjali Yadav*, Christina Giannoula†, Bahar Asgari*

* University of Maryland, College Park

† Max Planck Institute for Software Systems

This work is supported by **USA DoE** and **NSF**.

