AHAS : Adaptive Hybrid Accelerator for ScRNA

Shivaritha Sakthi Rengasamy

Date: 12/08/2025


MICHAEL JACKSON
Billie Jean
JEANS
GENES

The Biological Context

Genomic Analysis

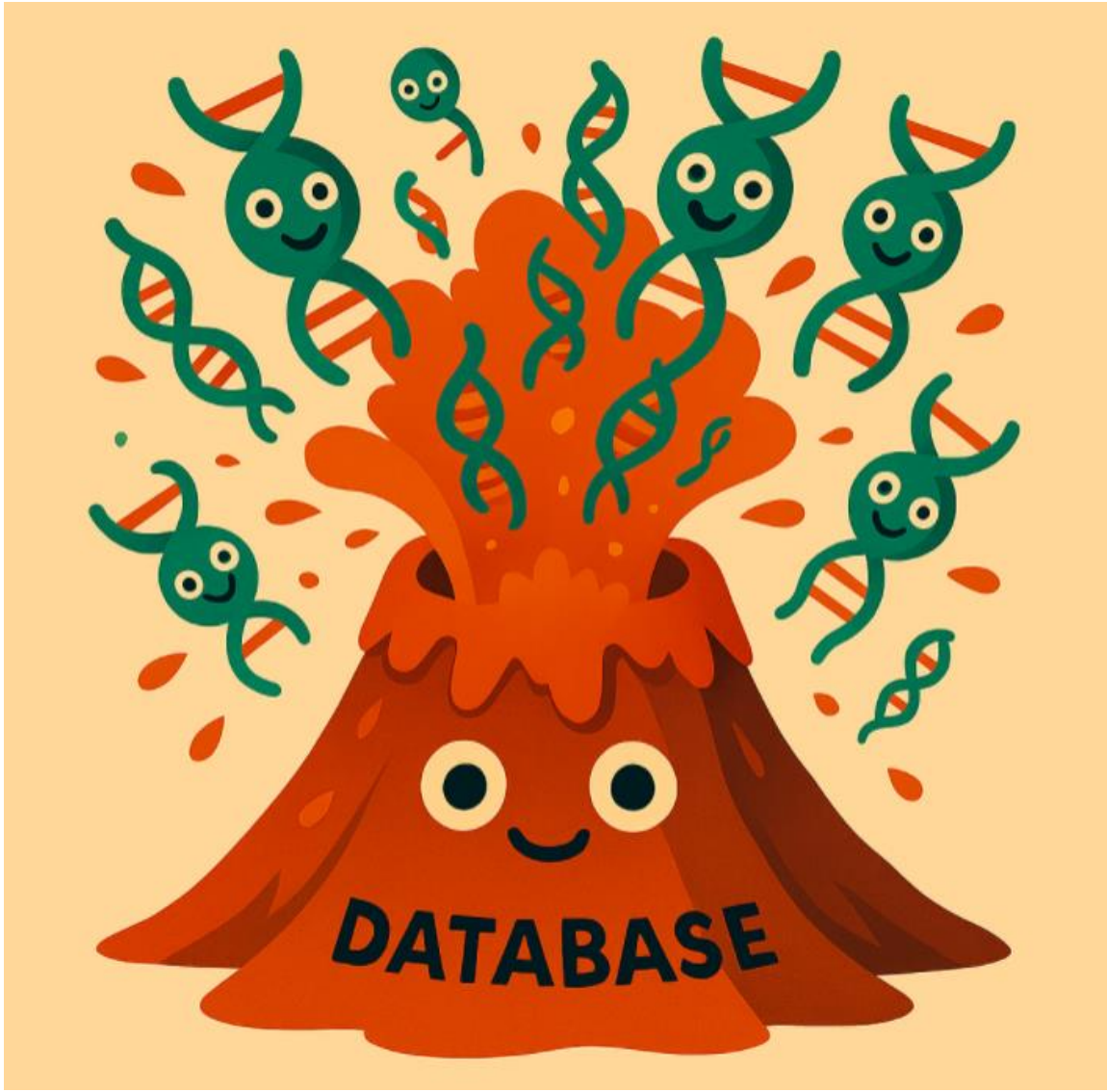
Decoding the blueprint of life to understand cellular identity.

Biological Fidelity

Identifying the **rare biological components** amidst the noise.

What Limits Genome Sequence Analysis?

- Significant data movement between storage, CPU, and accelerators.
- Every raw data transfer can amplify the volume of intermediate results.
- The database accumulates noisy, low-value data and grows much faster than the actual compute needs.



PROPOSED SOLUTION

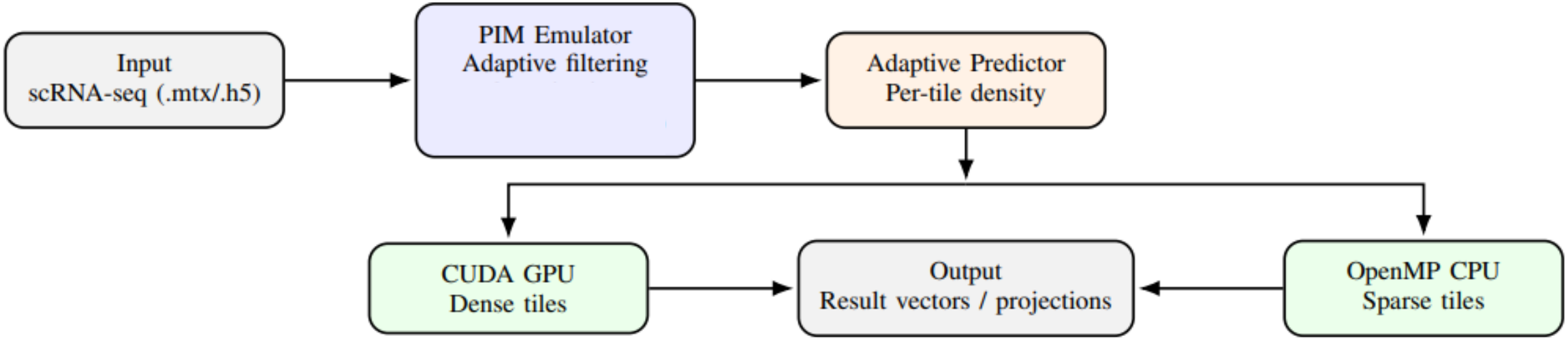


Fig. 1. Hybrid Adaptive SpMM framework.

PIM FILTER

- Variance computation: computes variance per gene across all cells using np.var (two-pass: mean, then variance).
- Gene classification: identifies rare genes (top 10% variance), noise genes (bottom 10%), and housekeeping genes (high mean, low variance).
- Filtering: keeps only rare genes (highly variable genes) and saves them to a filtered H5 file.

PIM filtering cuts nonzeros by **about 40-85%** and rows by **around 90%**, so each dataset becomes much smaller before compute.

The Bottleneck: Data Movement

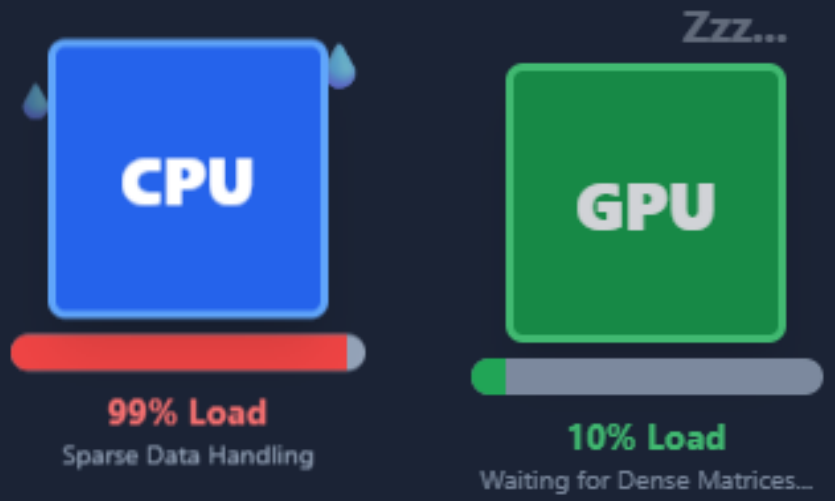
TARGET

Reduce I/O transfers between storage and CPU/GPU.

CHALLENGE

Retain biologically important genes.

Wait... that means the dataset is **SPARSE**, not dense?



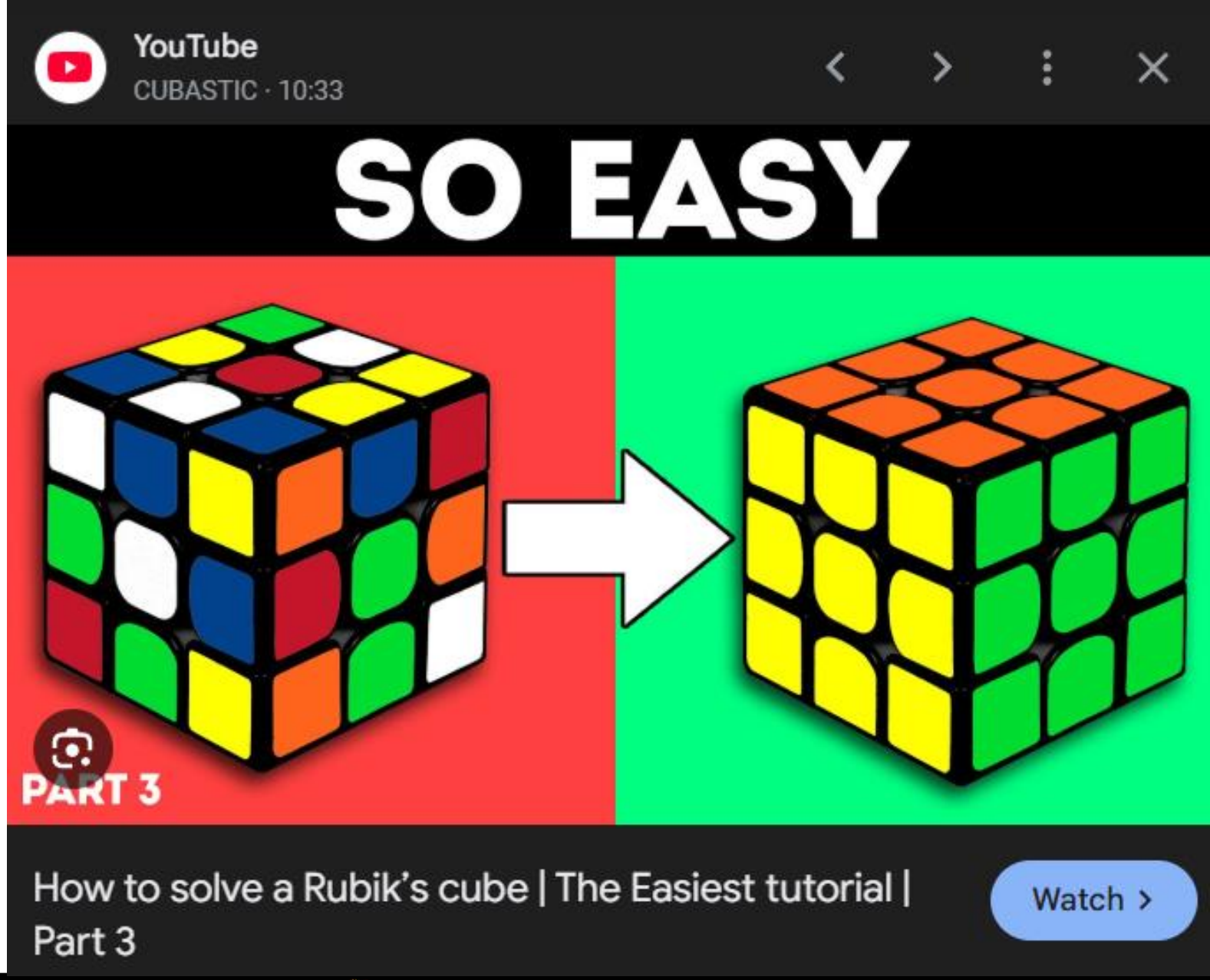
ARE TILES GPU FRIENDLY?

Dense tiles are permuted by sorting rows and columns in descending order by their non-zero counts

The code does this
in dense_perm_spmm_tile():

- Rows are sorted by nnz_per_row in descending order
- Columns are sorted by nnz_per_col in descending order
- The corresponding W weight matrix rows are permuted to match

This groups dense regions together so the GPU can access memory more efficiently.



Observation

- A sanity check is done between CUDA output vs reference to catch errors.
- IEEE 754: Floating-point addition is not associative; order changes results.
- Example: `(a+b)+c` can differ from `a+(b+c)` due to rounding.
- Parallel impact: GPU/parallel code often sums in different orders than CPU.
- Expected drift: Small numeric deltas are normal; large ones indicate issues.
- Tolerances used: `rtol = 1e-3` (0.1% relative), `atol = 1e-5` (absolute).
- Match rule: Values match if `|a-b| <= atol + rtol * max(|a|, |b|)`.
- Goal: Ignore harmless floating-point noise; flag true computation errors.

DATASET

Dataset	Dataset Name	Dimensions (rows × cols)	NNZ	Total Elements	Density	Sparsity
1	10k Mouse Splenocytes (5p, GEM-X)	33,696 × 9,263	26,407,826	312,123,648	8.46%	91.54%
2	10k Human DTC Melanoma (NextGEM 5p)	38,606 × 6,704	14,971,913	258,814,624	5.78%	94.22%
3	PBMC from Healthy Donor, Granulocytes Removed (≈3k cells)	134,920 × 2,711	24,511,186	365,978,120	6.70%	93.30%
4	30k A549 Lung Carcinoma Cells, CRISPR Pool, Multiplexed	36,706 × 2,979	16,028,697	109,428,174	14.64%	85.36%

DATASET 3 – 2.1x Faster

Metric	Baseline	tilepredpermspmm	Improvement
Compute time	137.9 ms	66.1 ms	2.1x faster
Input nnz	24.5 M	3.4 M	86% reduction
Input rows	134,920	3,660	97% reduction
Tiles	-	2,494	-
Dense tiles	-	43 (1.7%)	-
Sparse tiles	-	2,451 (98.3%)	-

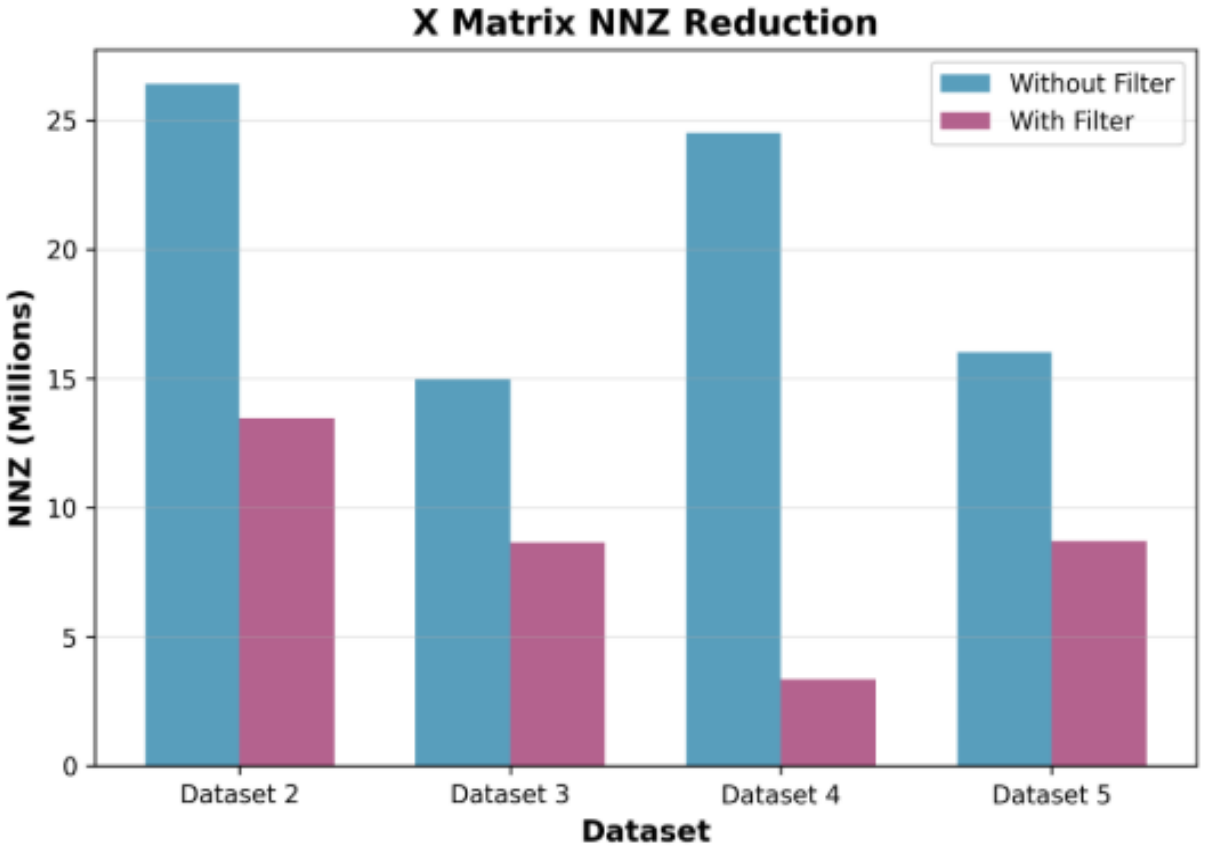
DATASET 4 – GPU overload

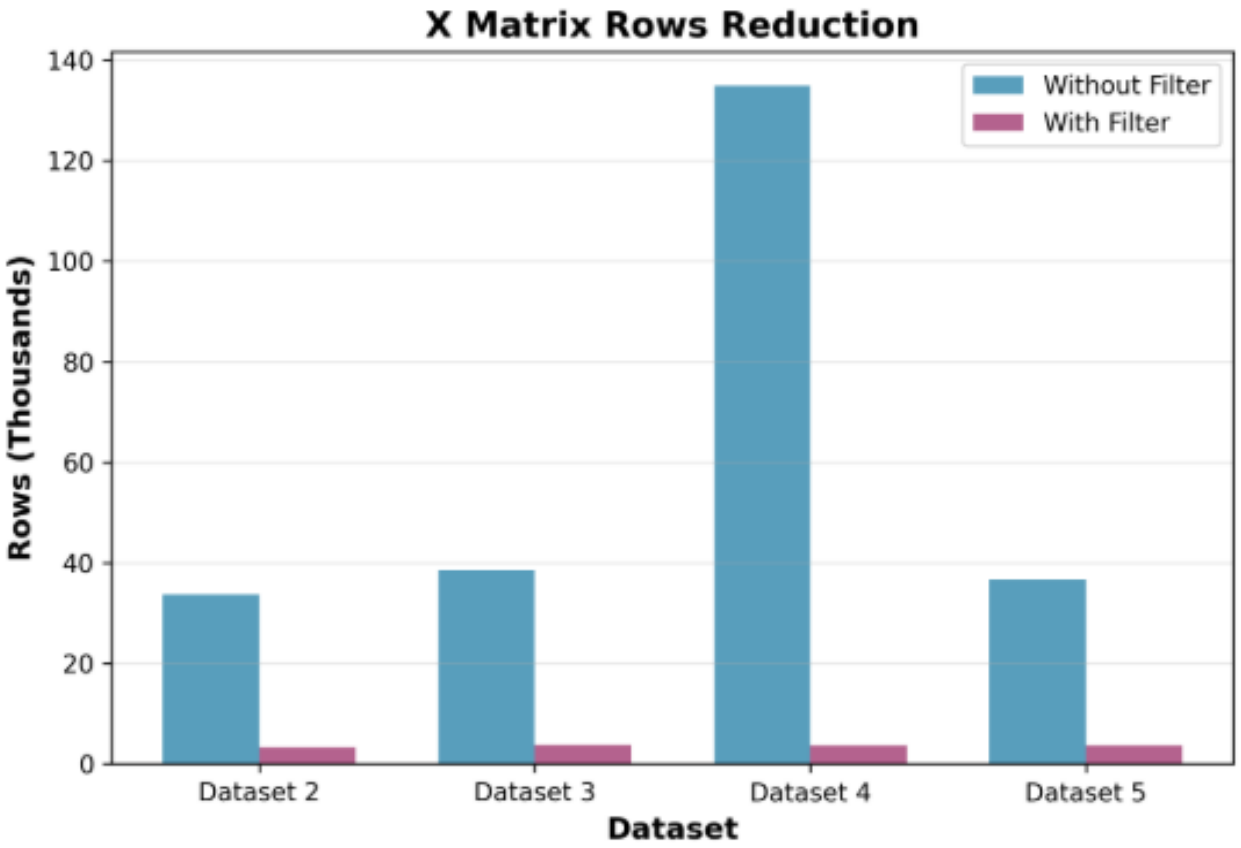
Metric	Baseline	tilepredpermspmm	Change
Compute time	53.6 ms	157.8 ms	2.9x slower
Input nnz	16.0 M	8.7 M	46% reduction
Input rows	36,706	3,660	90% reduction
Tiles	-	2,726	-
Dense tiles	-	2,726 (100%)	-
Sparse tiles	-	0 (0%)	-
Matrix density	-	0.799	-

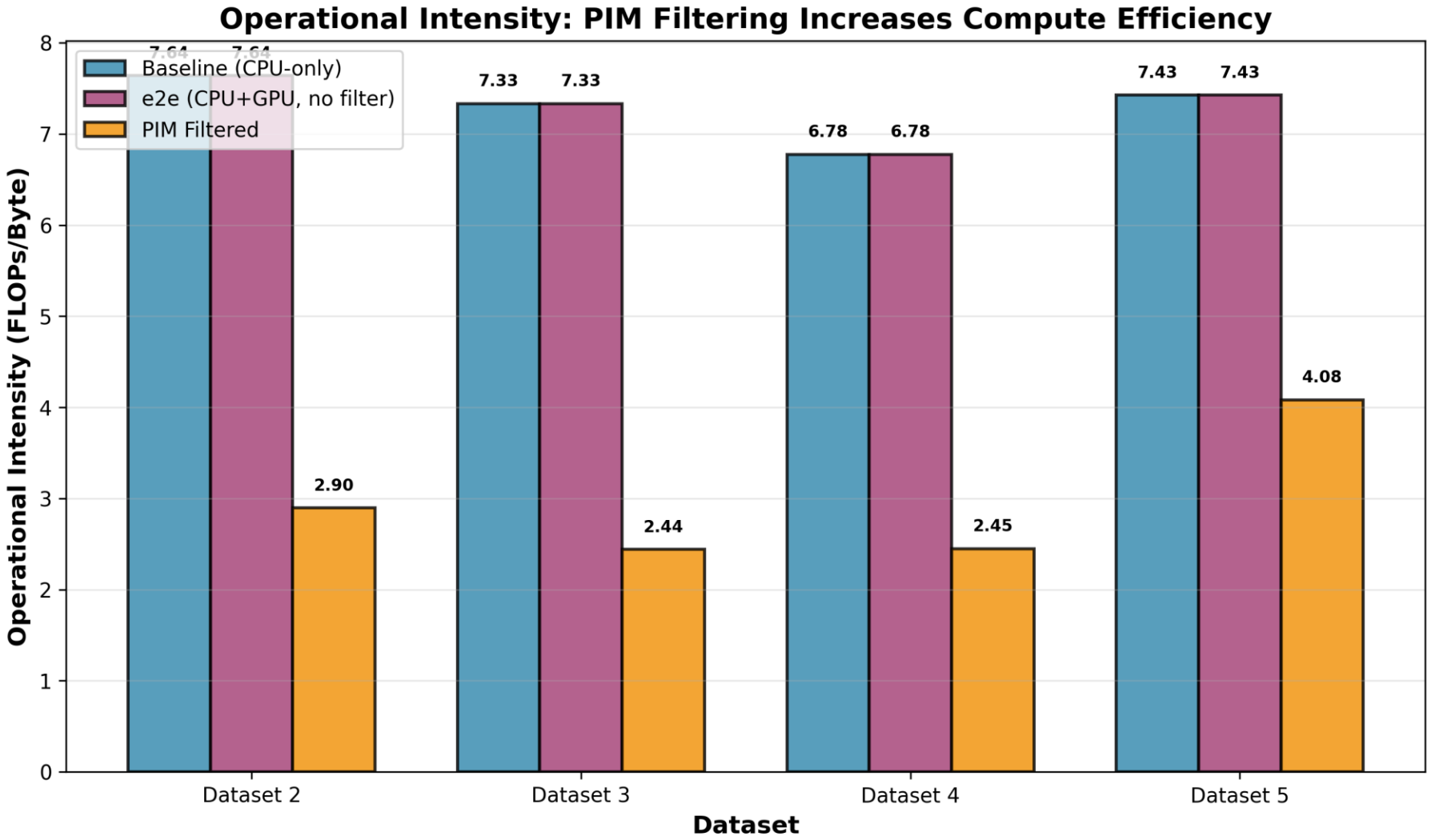
GPU USAGE

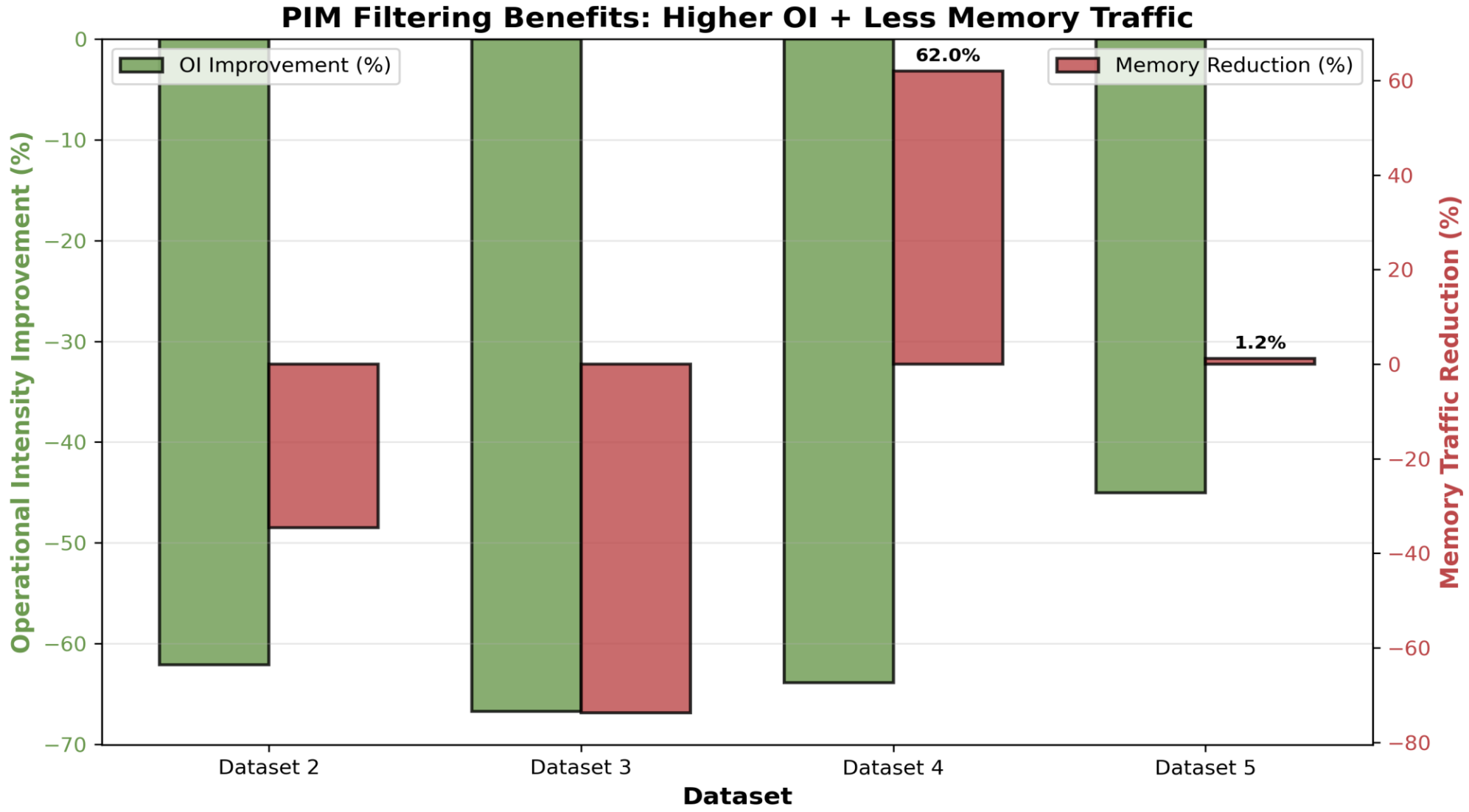
Dataset	Dense Tiles	GPU Workload	GPU Benefit
5	2,726 (100%)	High	Limited by launch overhead
2	1,049 (13.6%)	Moderate	Coordination overhead
3	105 (1.6%)	Low	Minimal
4	43 (1.7%)	Low	Minimal

X matrix Cells x Genes

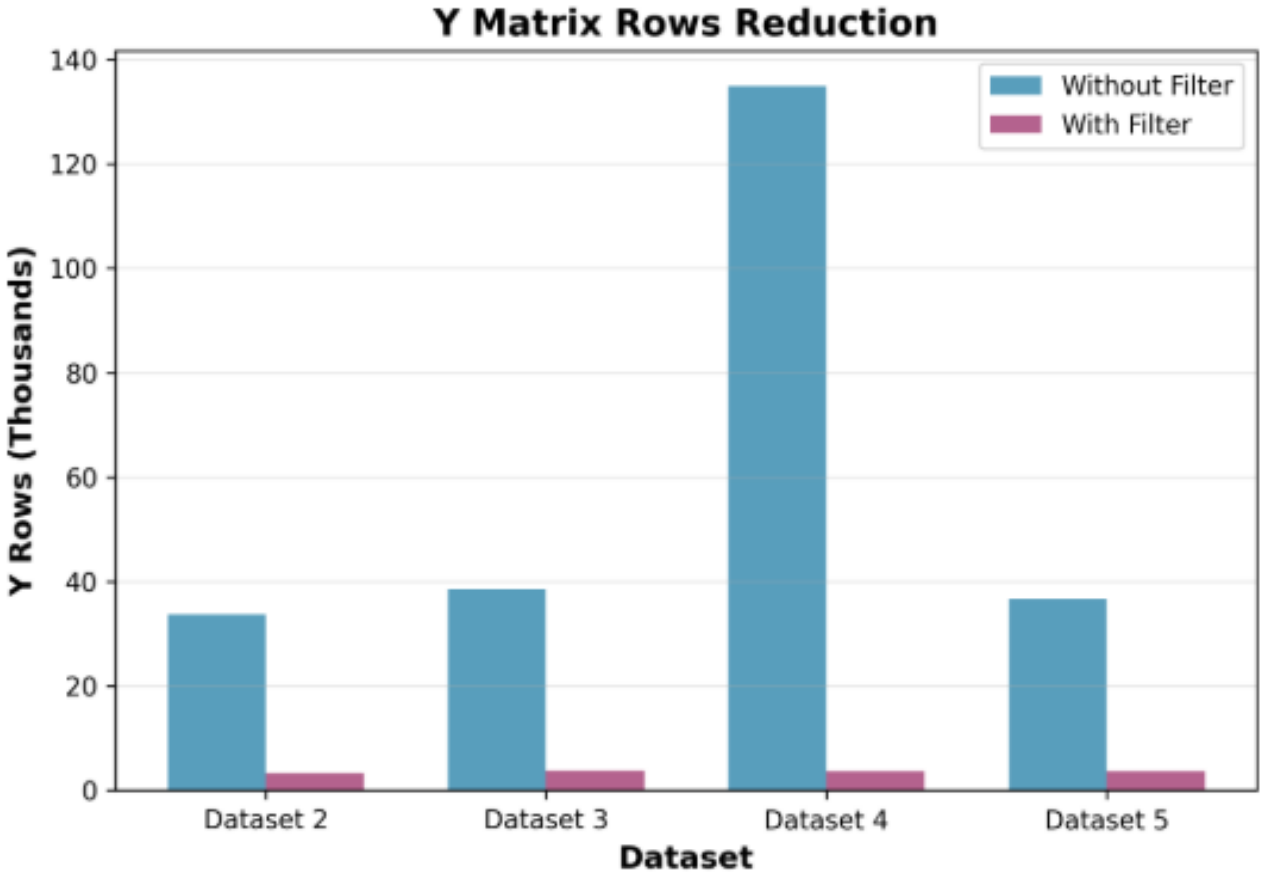


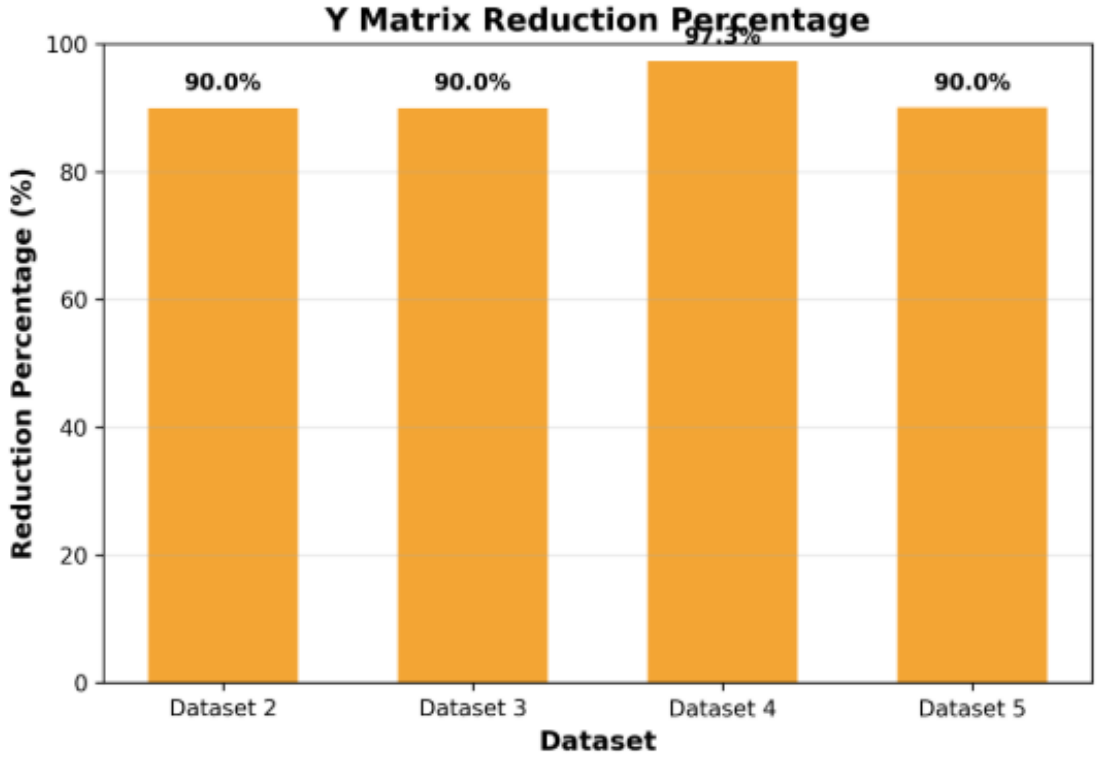


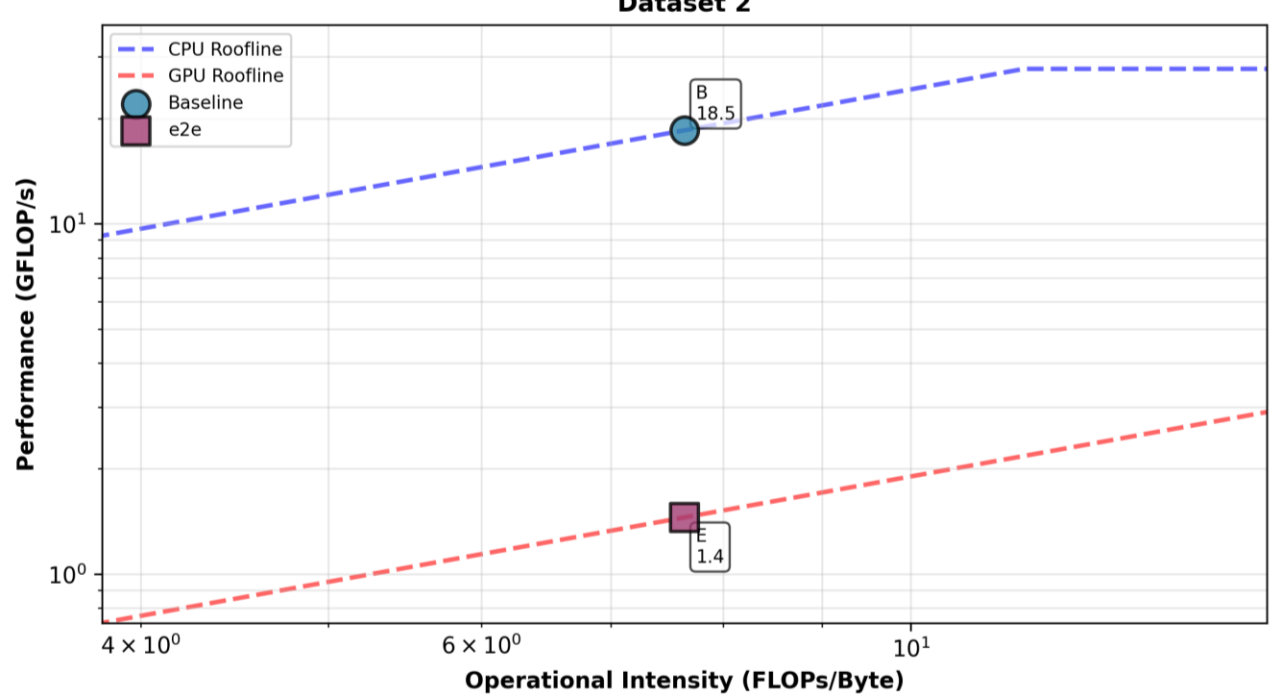

Operational Intensity: PIM Filtering Increases Compute Efficiency
Baseline (CPU-only)
e2e (CPU+GPU, no filter)
PIM Filtered
Operational Intensity (FLOPs/Byte)
Dataset
Dataset 2
Dataset 3
Dataset 4
Dataset 5
7.64
7.64
2.90
7.33
7.33
2.44
6.78
6.78
2.45
7.43
7.43
4.08
0
1
2
3
4
5
6
7
8

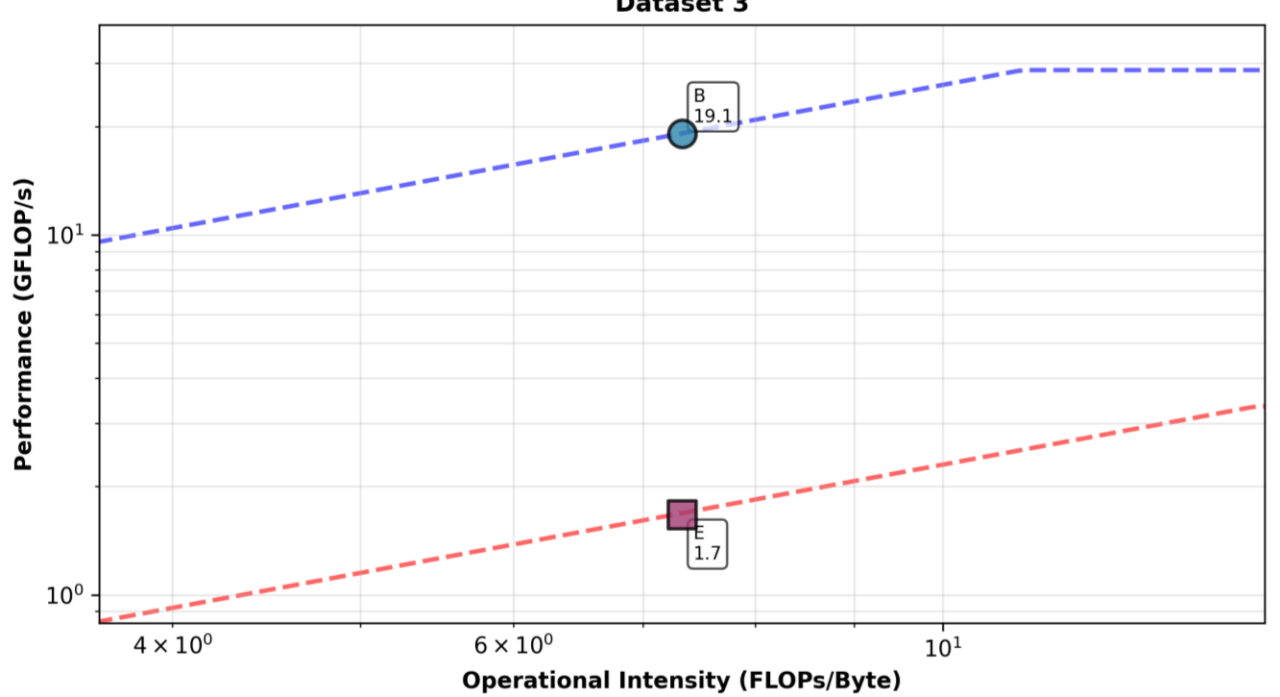

PIM Filtering Benefits: Higher OI + Less Memory Traffic
OI Improvement (%)
Memory Reduction (%)
62.0%
1.2%
Operational Intensity Improvement (%)
Memory Traffic Reduction (%)
0
−10
−20
−30
−40
−50
−60
−70
60
40
20
0
−20
−40
−60
−80
Dataset 2
Dataset 3
Dataset 4
Dataset 5
Dataset

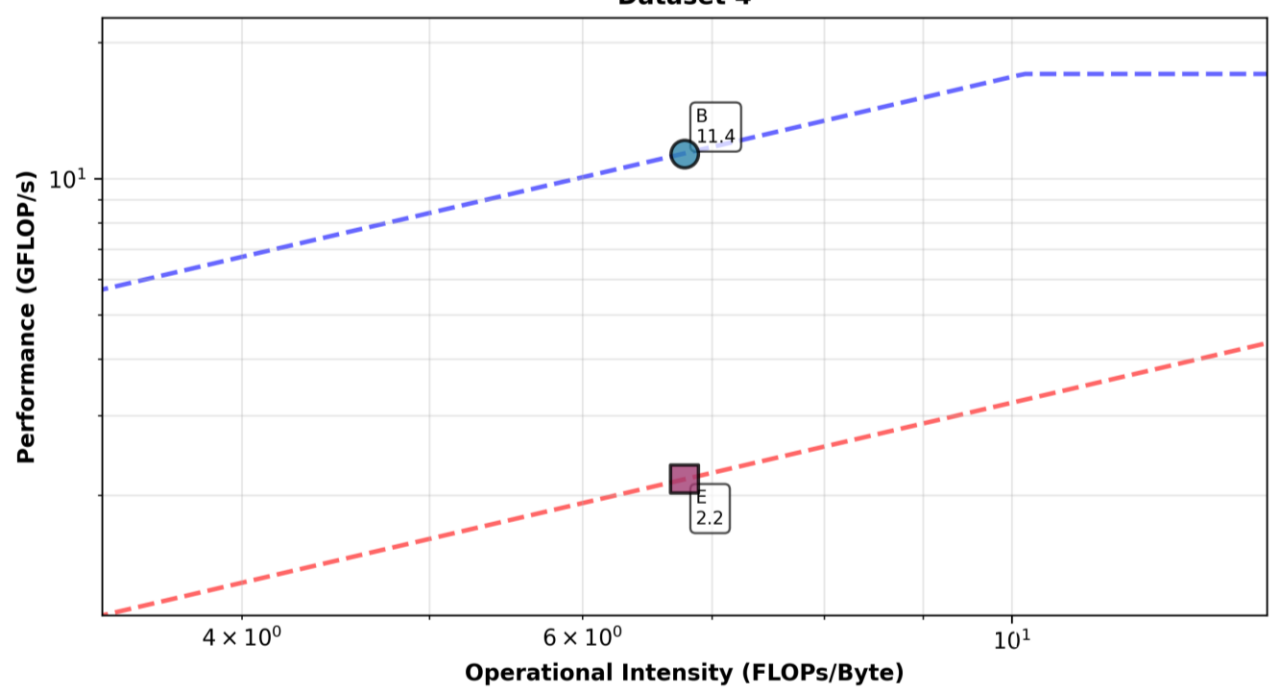
OUTPUT MATRIX Y = X * W

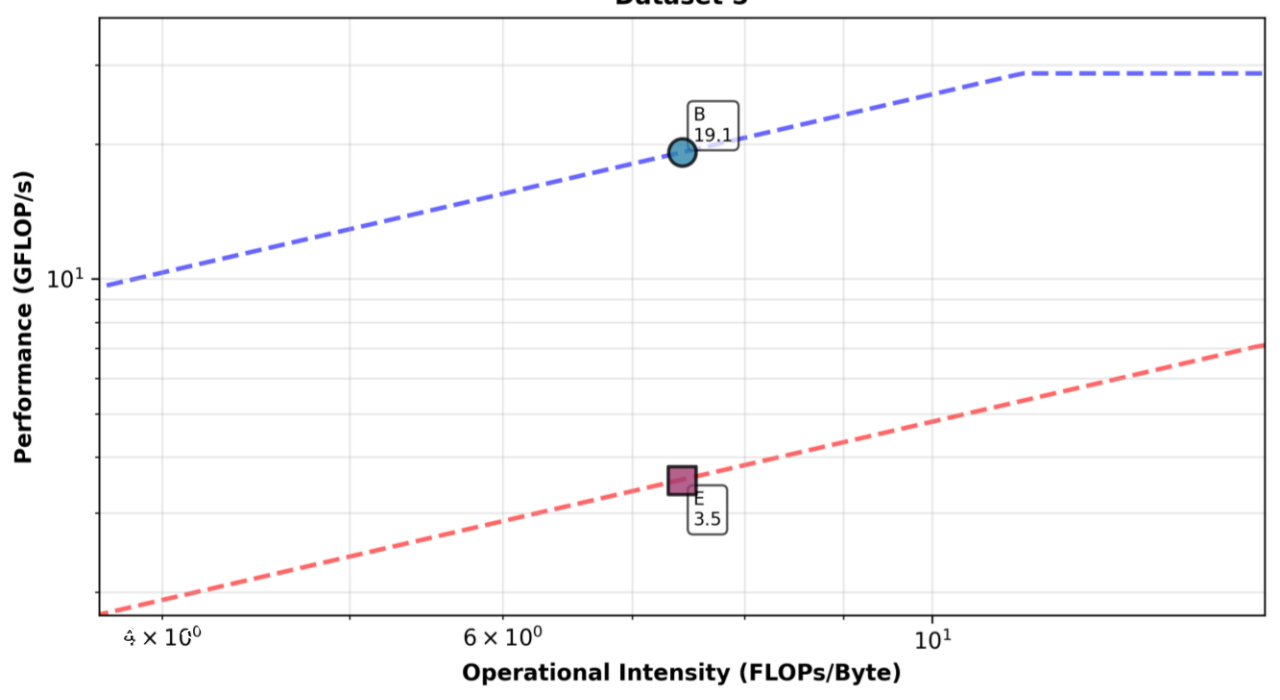


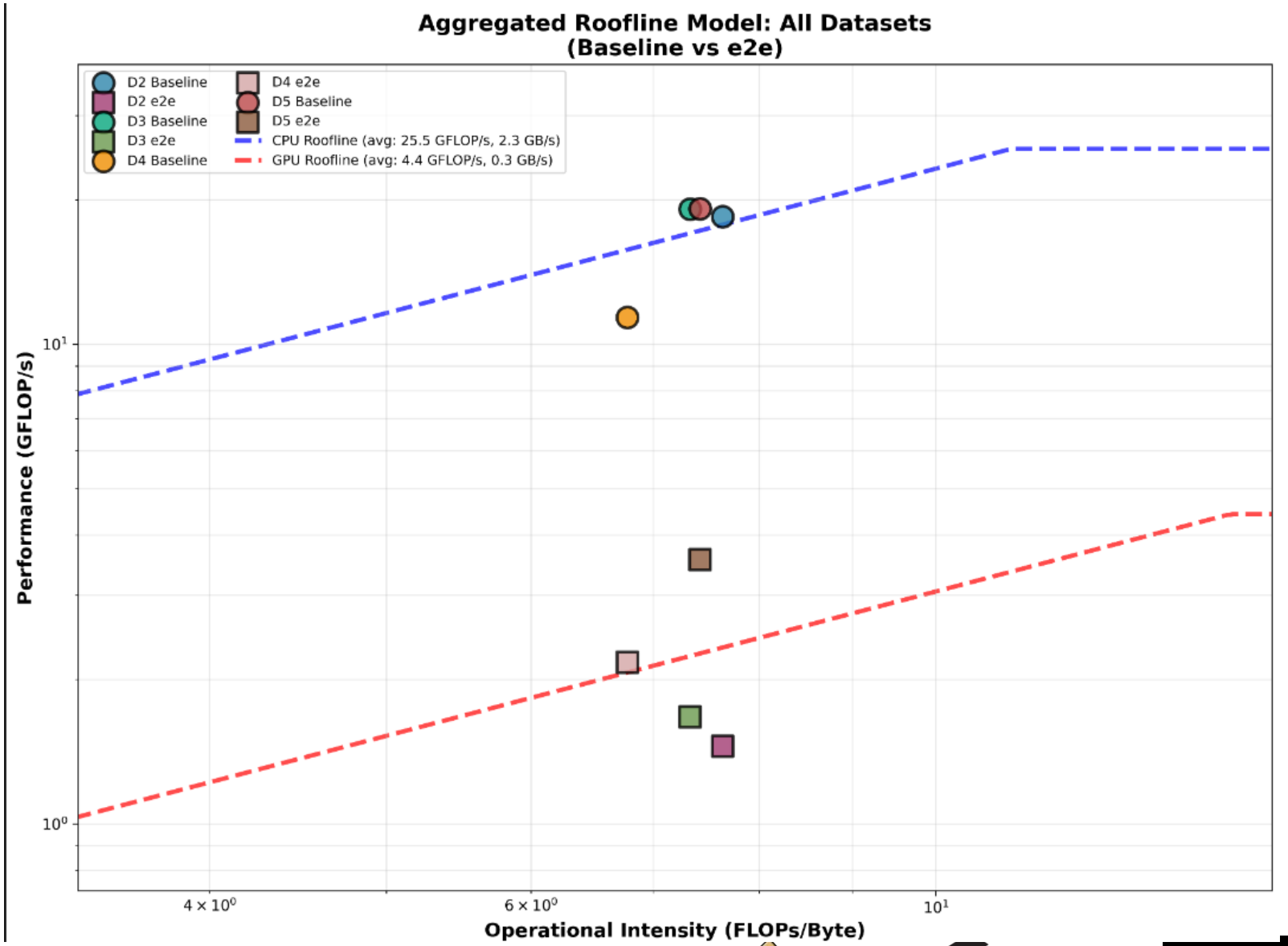



Dataset 2
CPU Roofline
GPU Roofline
Baseline
e2e
B
18.5
E
1.4
Performance (GFLOP/s)
Operational Intensity (FLOPs/Byte)


Dataset 3
B
19.1
E
1.7
Performance (GFLOP/s)
Operational Intensity (FLOPs/Byte)


Dataset 4
B
11.4
E
2.2
Performance (GFLOP/s)
Operational Intensity (FLOPs/Byte)


Dataset 5
B
19.1
E
3.5
Performance (GFLOP/s)
Operational Intensity (FLOPs/Byte)


Aggregated Roofline Model: All Datasets
(Baseline vs e2e)
D2 Baseline
D2 e2e
D3 Baseline
D3 e2e
D4 Baseline
D4 e2e
D5 Baseline
D5 e2e
CPU Roofline (avg: 25.5 GFLOP/s, 2.3 GB/s)
GPU Roofline (avg: 4.4 GFLOP/s, 0.3 GB/s)
Performance (GFLOP/s)
10^1
10^0
4×10^0
6×10^0
10^1
Operational Intensity (FLOPs/Byte)

Contributions

- Demonstrates PIM filtering effectiveness with 42–86% nnz reduction across datasets.
- Provides a performance characterization.
- Reveals hybrid execution trade-offs between CPU and GPU workloads and coordination costs.

Future Work

- Adaptive tile sizing to use:
  - larger tiles for dense regions
  - smaller ones for sparse regions.
- Quantization of data.
- Look into structured sparsity
- GPU batching to aggregate multiple dense tiles into single GPU calls.
- Dynamic threshold tuning to adjust dense/sparse classification per dataset.
- Detailed overhead profiling to separately measure tiling, permutation, and transfer costs.

THANK YOU!!!