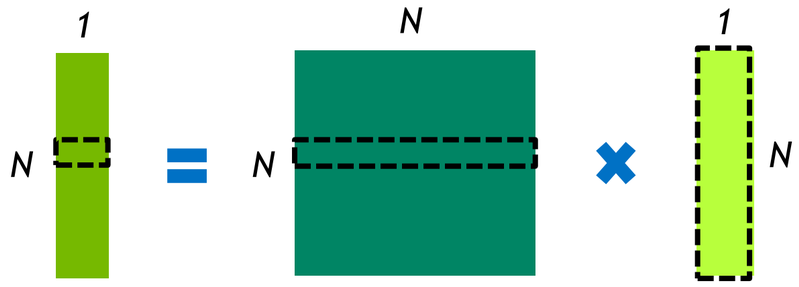
COMMUNITY-BASED MATRIX REORDERING FOR SPARSE LINEAR ALGEBRA OPTIMIZATION

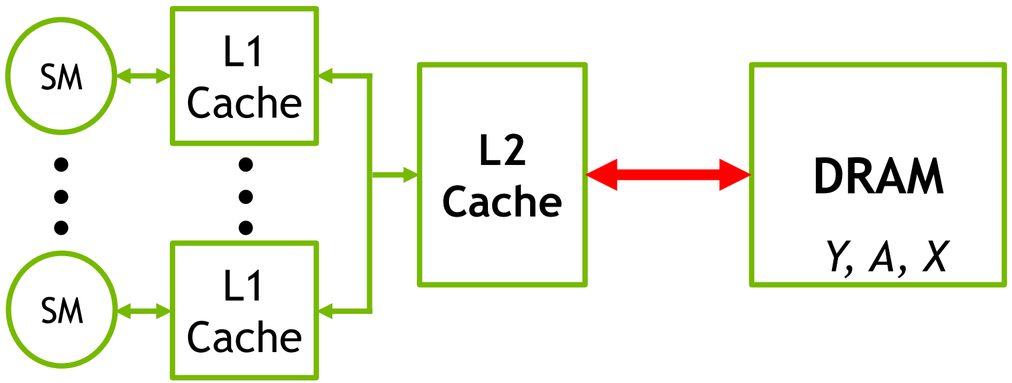
Vignesh Balaji Neal Crago Aamer Jaleel Stephen W. Keckler

PROBLEM – SUBOPTIMAL SPARSE LINEAR ALGEBRA PERFORMANCE

PROBLEM – SUBOPTIMAL SPARSE LINEAR ALGEBRA PERFORMANCE

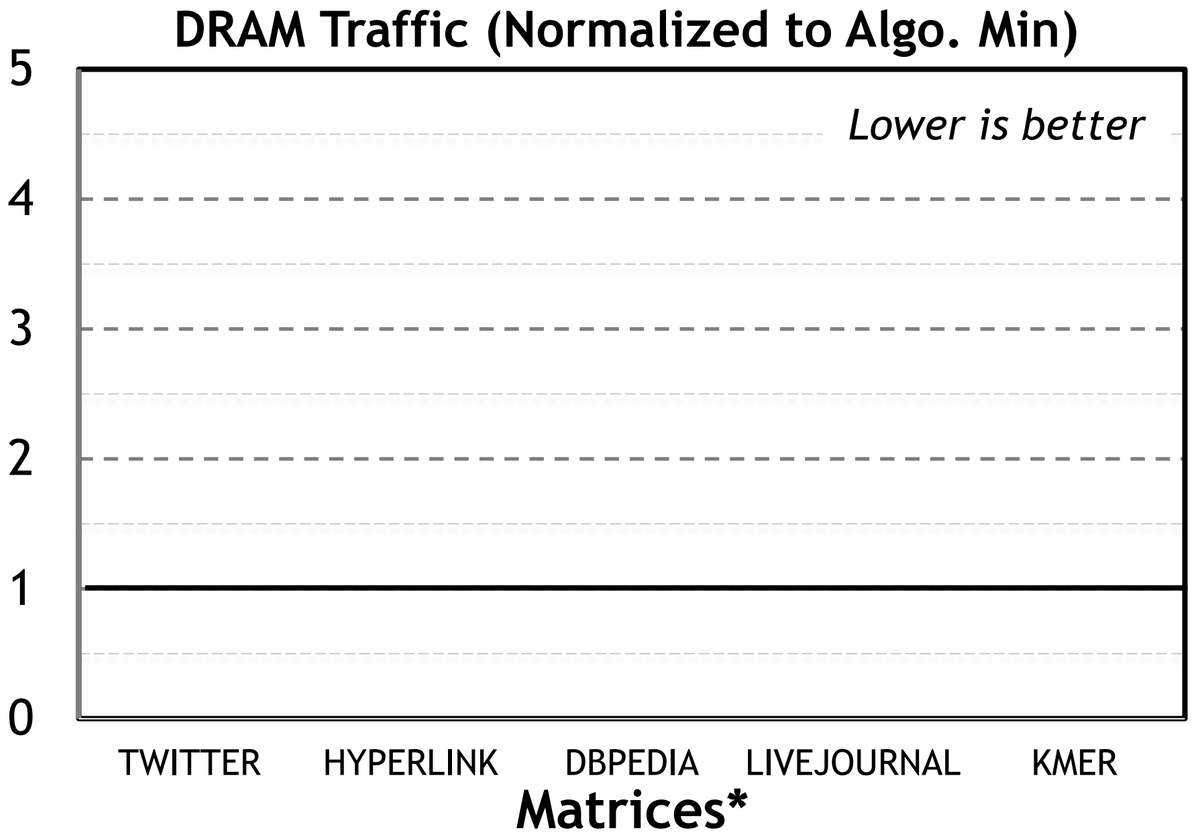


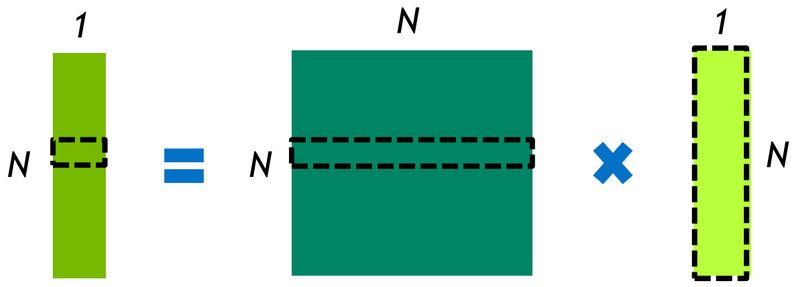
Sparse Matrix Vector Multiplication (SpMV)



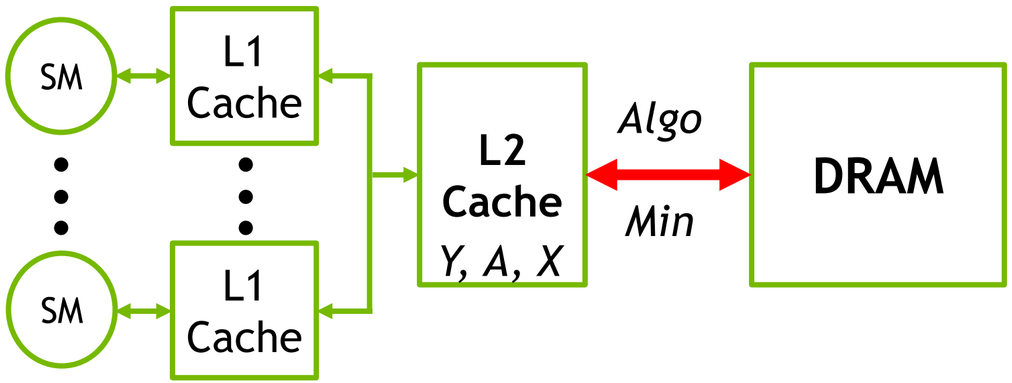
PROBLEM – SUBOPTIMAL SPARSE LINEAR ALGEBRA PERFORMANCE

Characterization on NVIDIA A6000 GPU





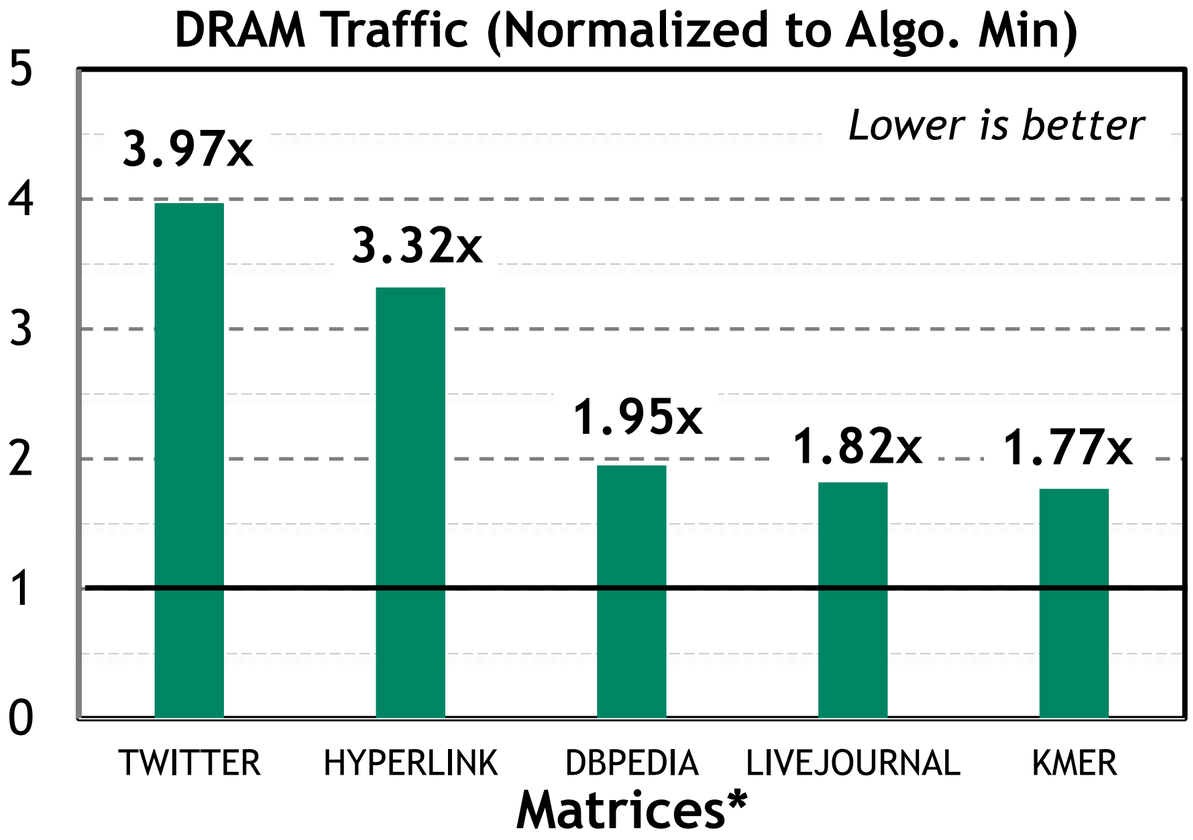
Sparse Matrix Vector Multiplication (SpMV)

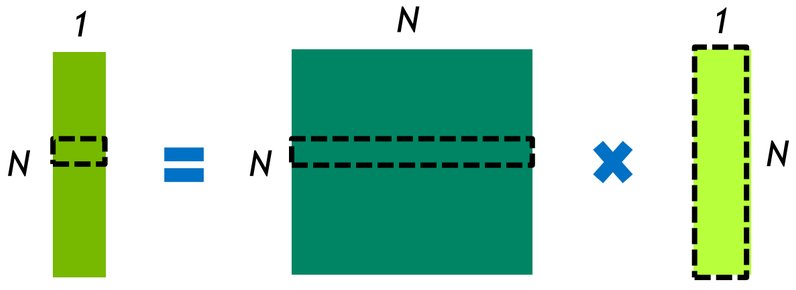


**Subset of matrices evaluated*

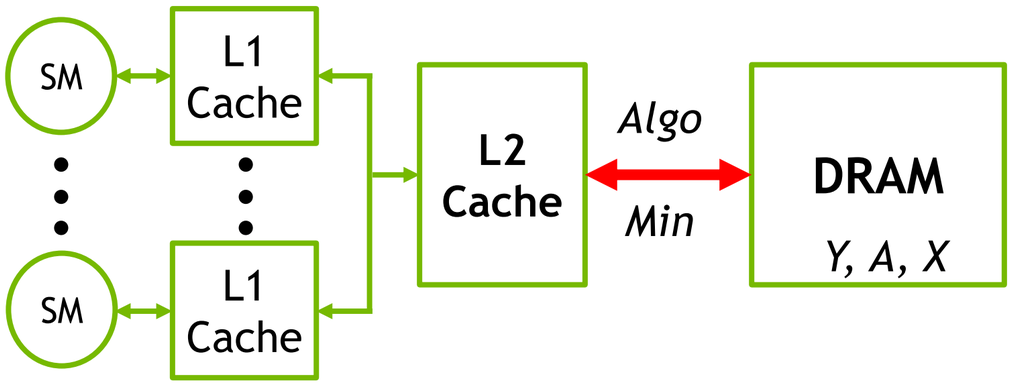
PROBLEM – SUBOPTIMAL SPARSE LINEAR ALGEBRA PERFORMANCE

Characterization on NVIDIA A6000 GPU



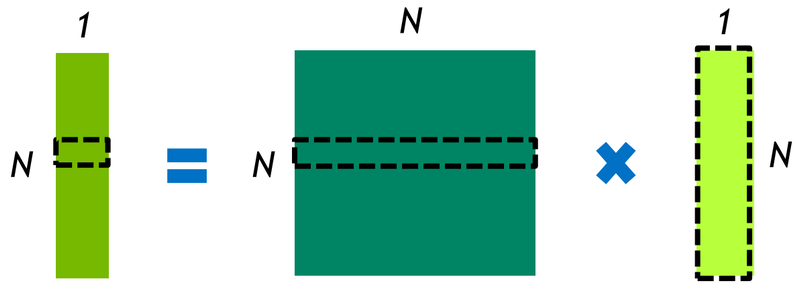


Sparse Matrix Vector Multiplication (SpMV)



**Subset of matrices evaluated*

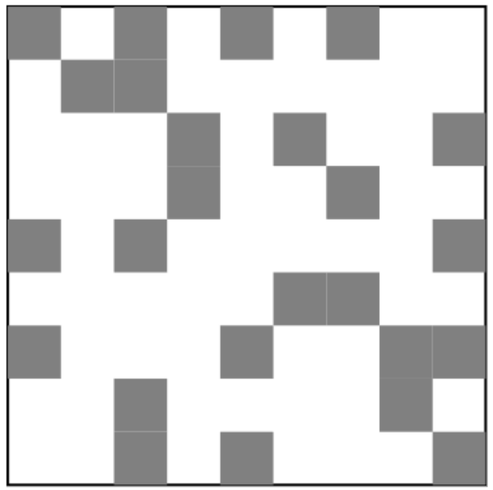
SOURCE OF POOR PERFORMANCE -- IRREGULARITY

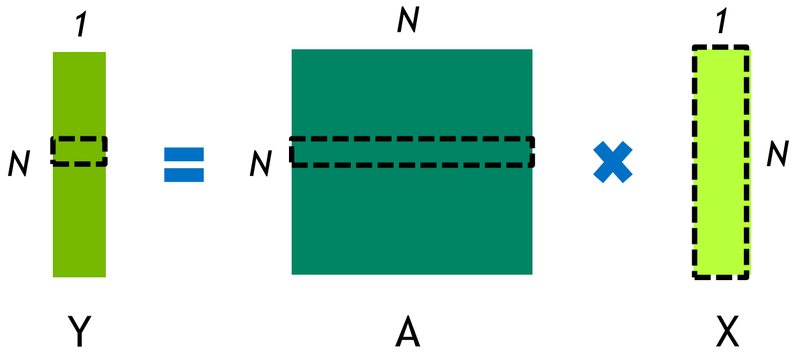


Sparse Matrix Vector Multiplication (SpMV)

SOURCE OF POOR PERFORMANCE -- IRREGULARITY

Typically, greater than 99% of entries are zeros

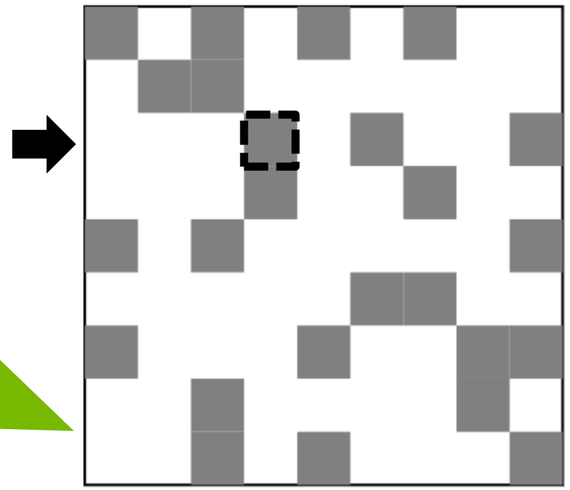


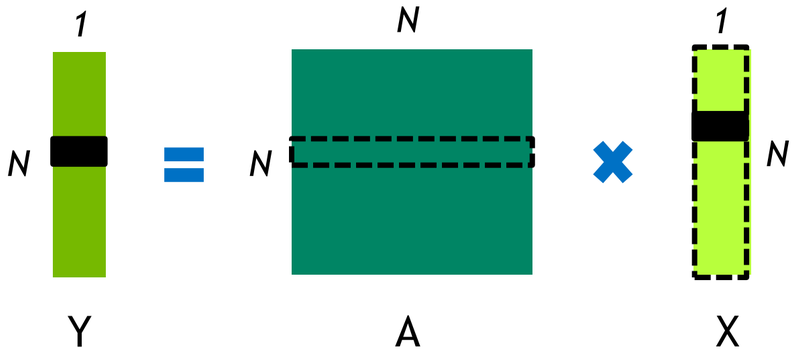


Sparse Matrix Vector Multiplication (SpMV)

SOURCE OF POOR PERFORMANCE -- IRREGULARITY

Typically, greater than 99% of entries are zeros

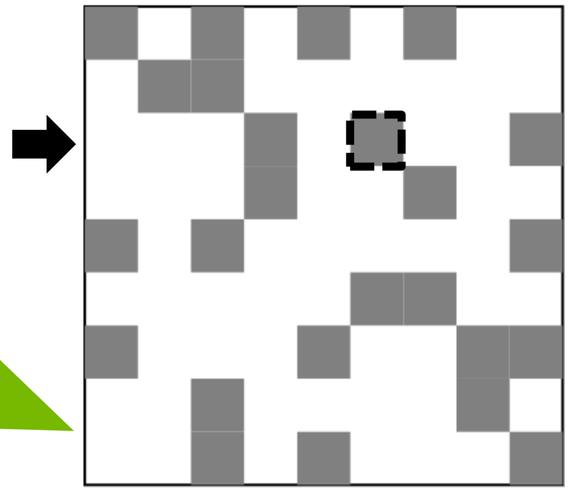


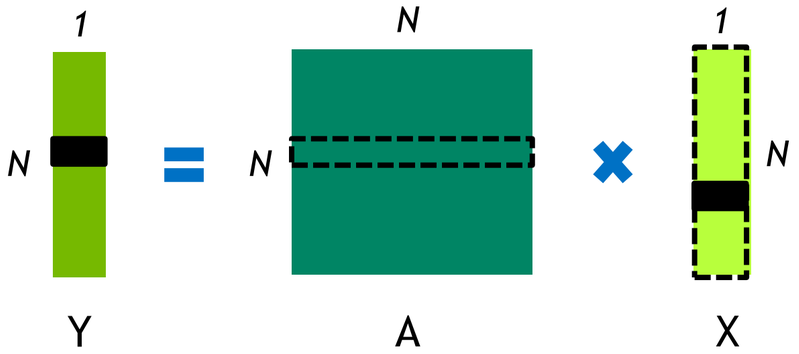


Sparse Matrix Vector Multiplication (SpMV)

SOURCE OF POOR PERFORMANCE -- IRREGULARITY

Typically, greater than 99% of entries are zeros

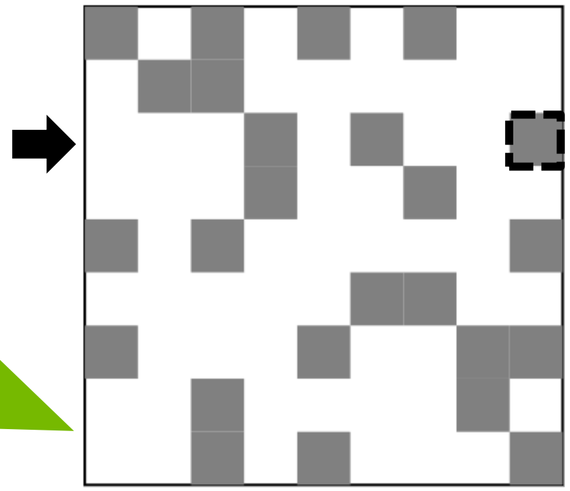


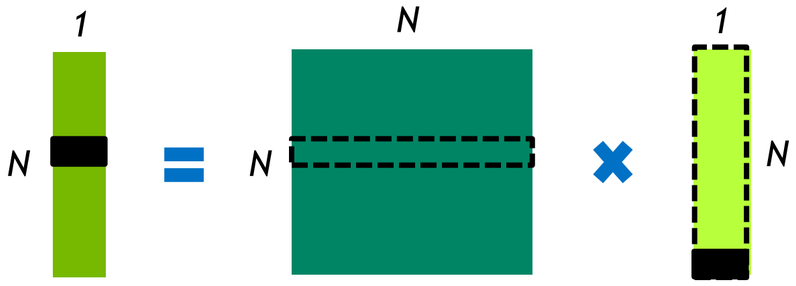


Sparse Matrix Vector Multiplication (SpMV)

SOURCE OF POOR PERFORMANCE -- IRREGULARITY

Typically, greater than 99% of entries are zeros

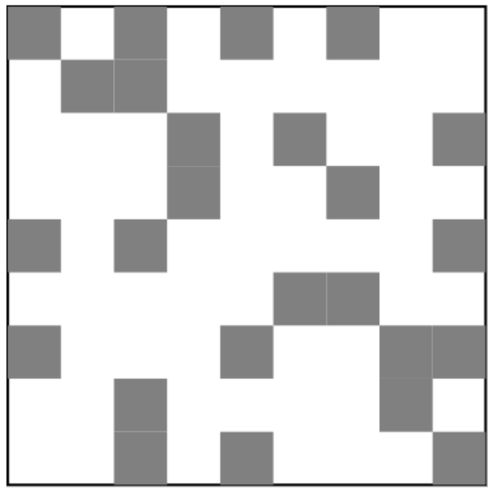




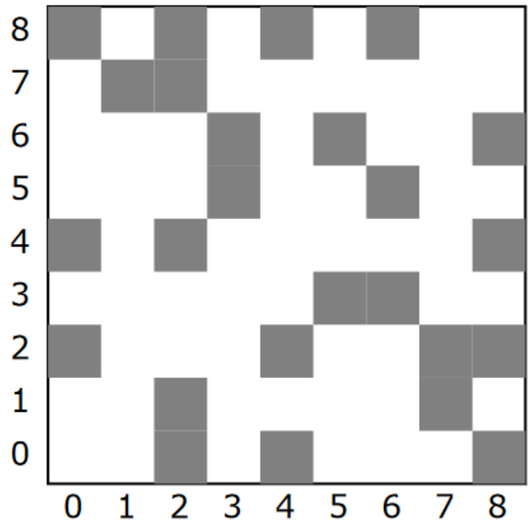
Sparse Matrix Vector Multiplication (SpMV)

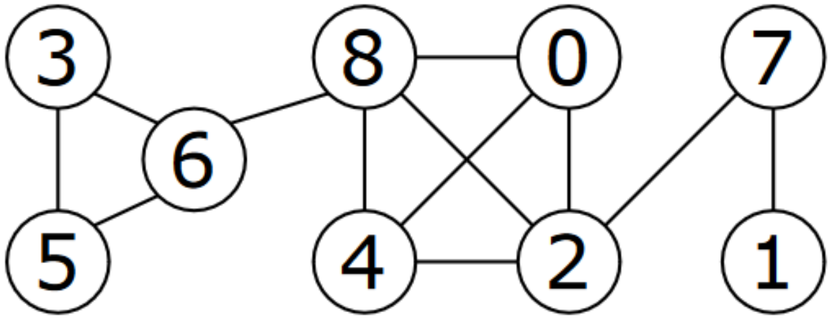
Compressed Representations of sparse matrices lead to fine-grained, irregular accesses to Input Vector (X)

IMPROVING REGULARITY WITH MATRIX REORDERING

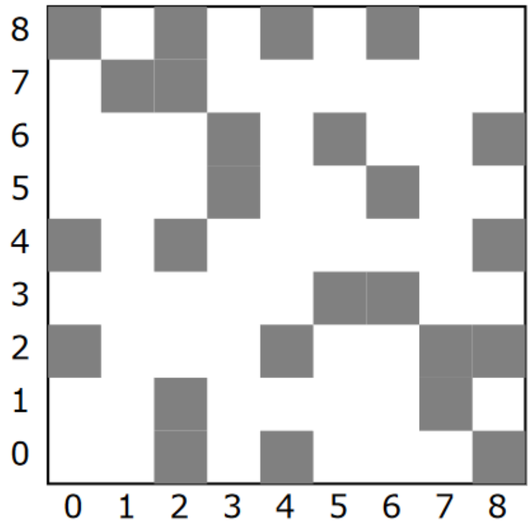


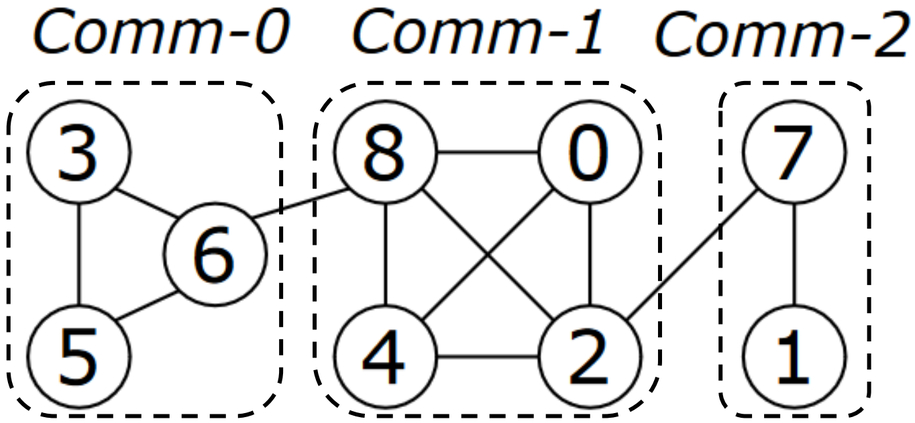
IMPROVING REGULARITY WITH MATRIX REORDERING





IMPROVING REGULARITY WITH MATRIX REORDERING

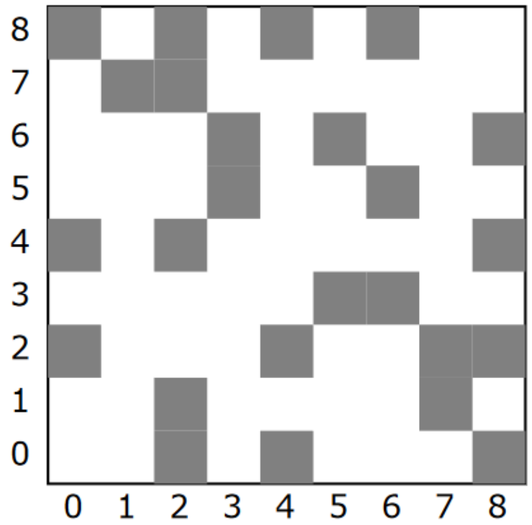


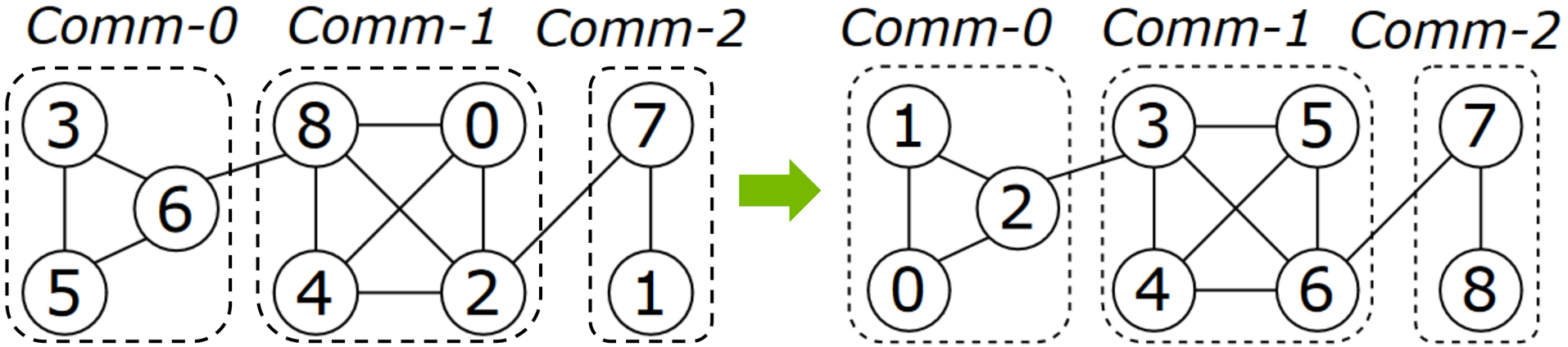


Many real-world networks exhibit community structure:

- Social Networks
- Web Crawls
- Biological Networks
- Knowledge graphs
- …

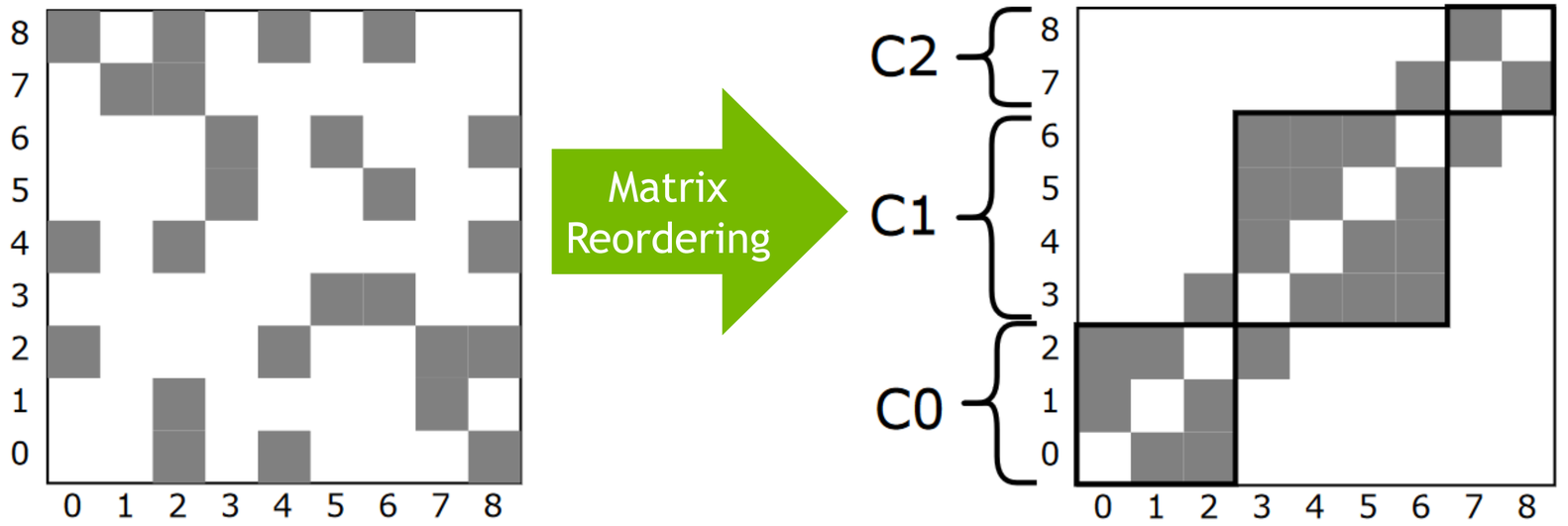
IMPROVING REGULARITY WITH MATRIX REORDERING

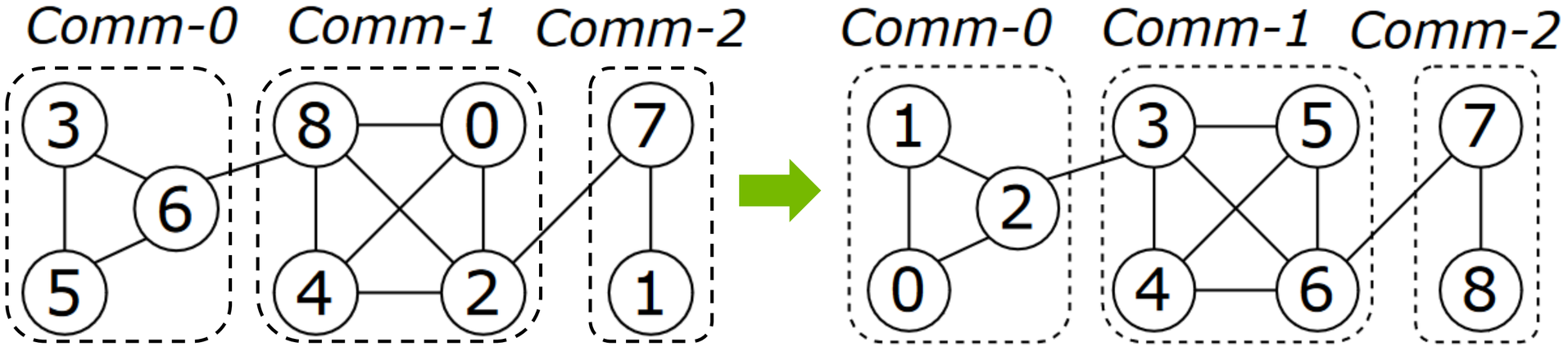




Relabel Matrix
Rows and Cols

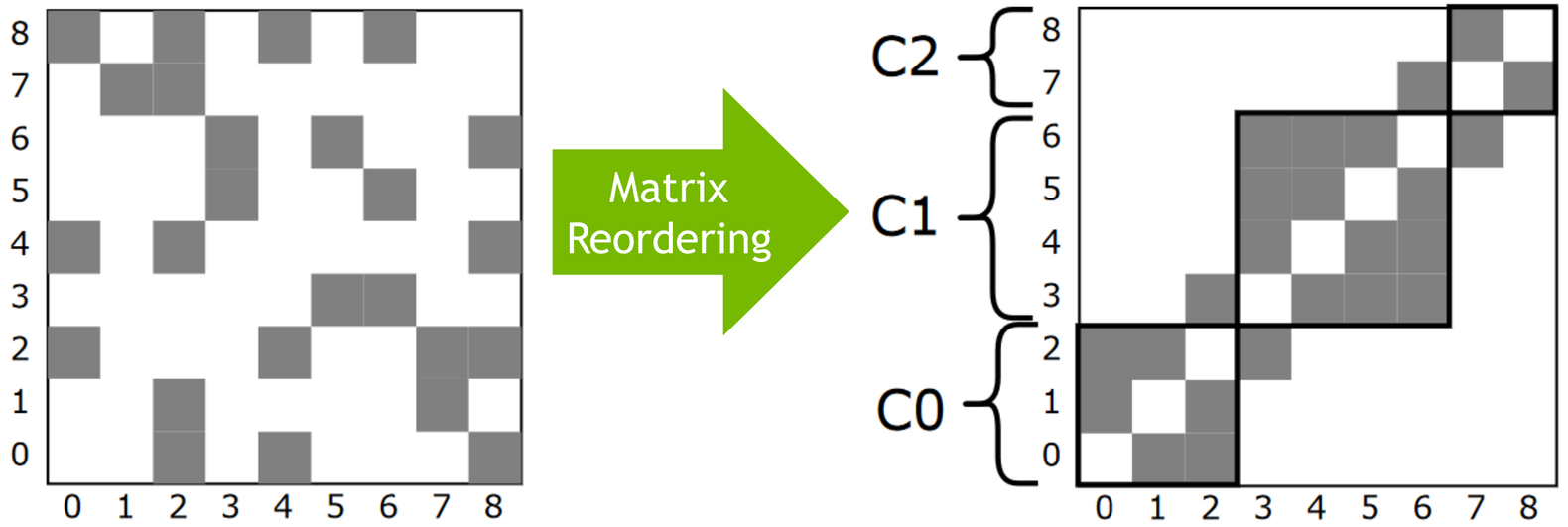
IMPROVING REGULARITY WITH MATRIX REORDERING

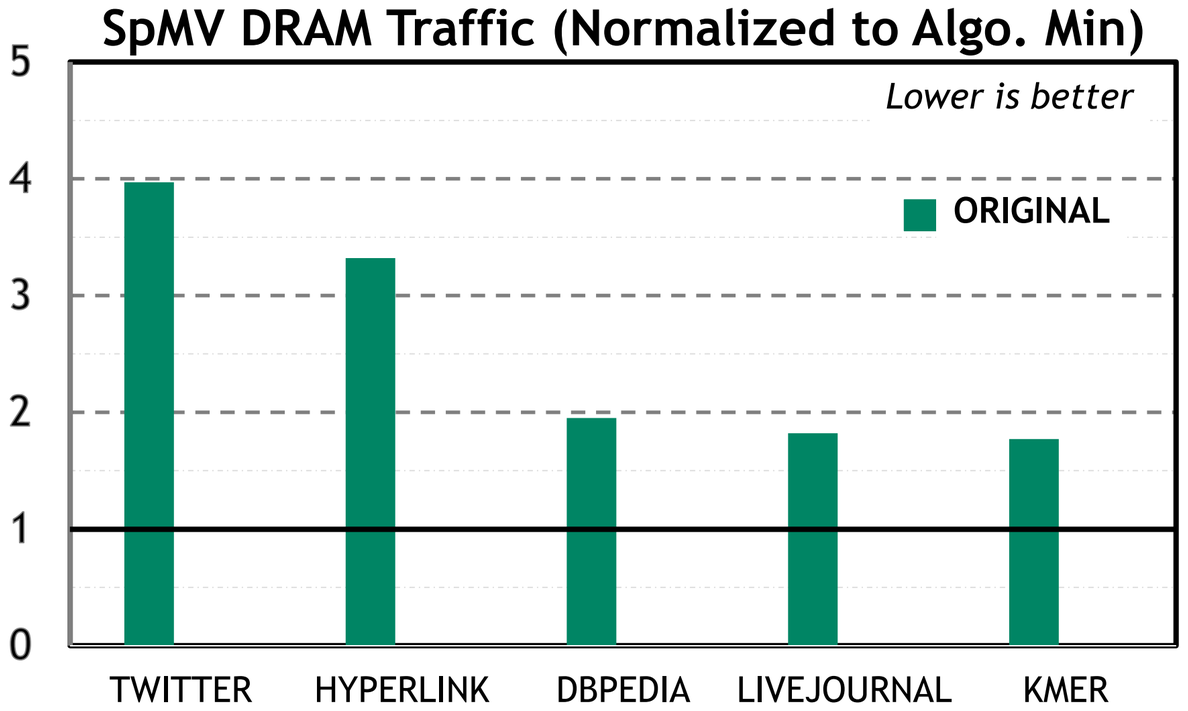




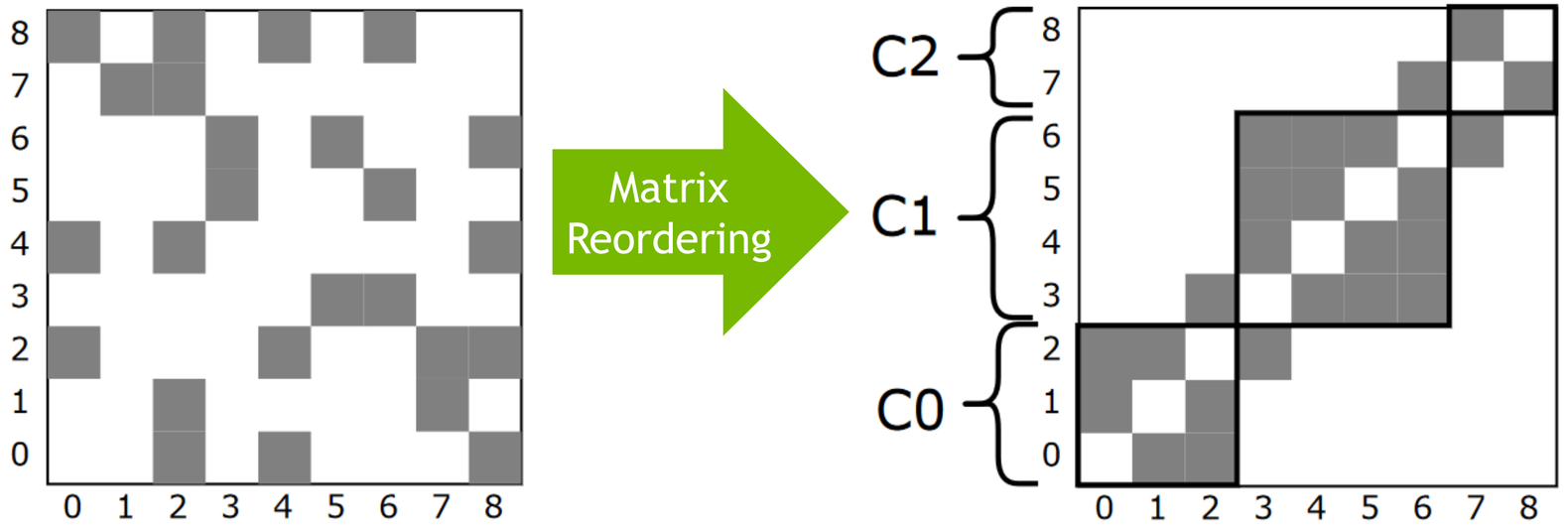
Relabel Matrix
Rows and Cols

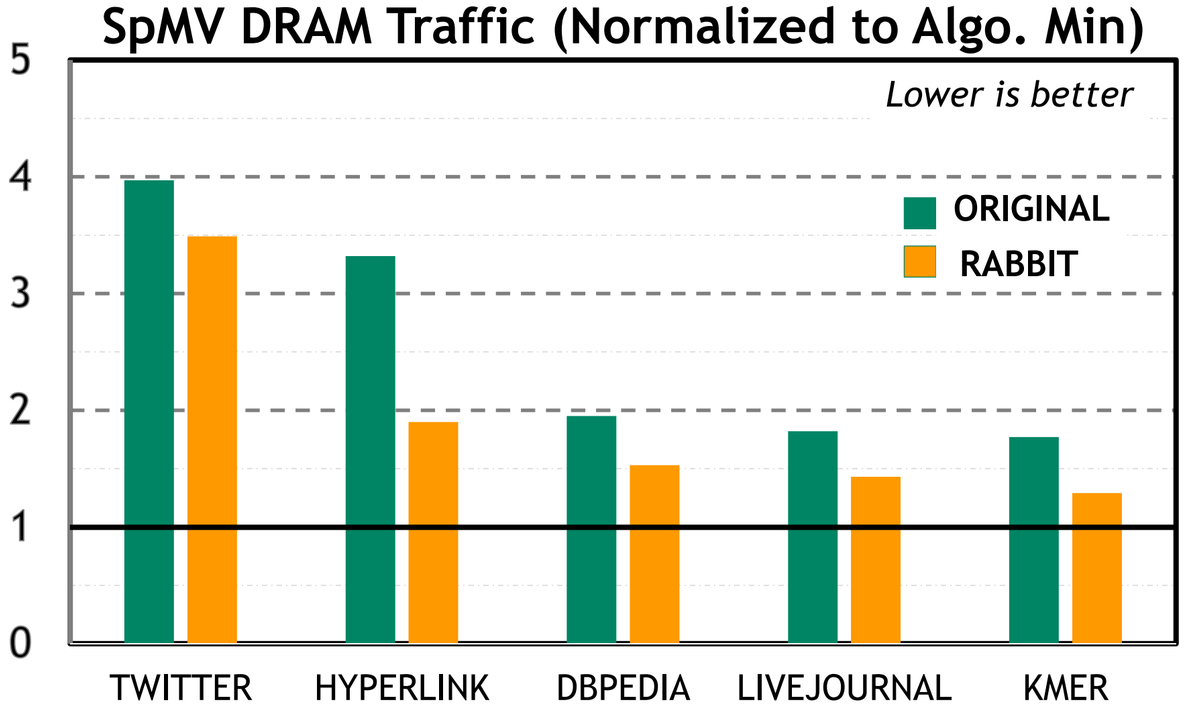
IMPROVING REGULARITY WITH MATRIX REORDERING





IMPROVING REGULARITY WITH MATRIX REORDERING





OUTLINE

- **Matrix Reordering Improves Locality** ✓
- Methodology for Evaluating Reordering Techniques
- Community-based matrix reordering (RABBIT) is best overall
- RABBIT++: Transformations to improve RABBIT

OUTLINE

- **Matrix Reordering Improves Locality** ✓
- Methodology for Evaluating Reordering Techniques ⬅
- Community-based matrix reordering (RABBIT) is best overall
- RABBIT++: Transformations to improve RABBIT

EVALUATION METHODOLOGY

HARDWARE: NVIDIA A6000 GPU

L2 Cache	DRAM Bandwidth	Mem Capacity
6MB	768GB/s	48GB

SOFTWARE: NVIDIA cuSPARSE library (v11.8)

EVALUATION METHODOLOGY

HARDWARE: NVIDIA A6000 GPU

L2 Cache	DRAM Bandwidth	Mem Capacity
6MB	768GB/s	48GB

SOFTWARE: NVIDIA cuSPARSE library (v11.8)

INPUTS: We select one matrix from *each* distinct group across 3 datasets

The KONECT Project

Web Data Commons

EVALUATION METHODOLOGY

HARDWARE: NVIDIA A6000 GPU

L2 Cache	DRAM Bandwidth	Mem Capacity
6MB	768GB/s	48GB

SOFTWARE: NVIDIA cuSPARSE library (v11.8)

INPUTS: We select one matrix from *each* distinct group across 3 datasets

The KONECT Project

Web Data Commons

Our final input set comprises of 50 matrices spanning diverse domains

OUTLINE

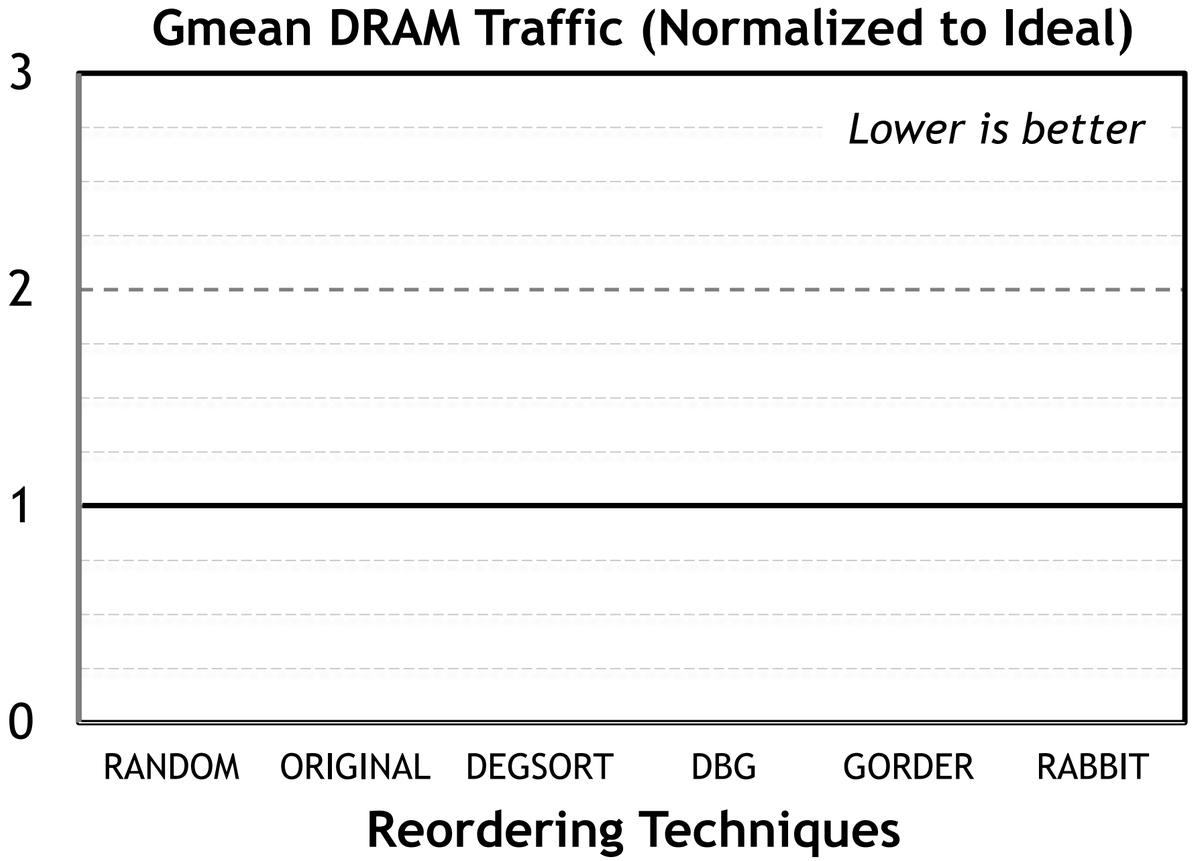
- **Matrix Reordering Improves Performance** ✓
- **Methodology for Evaluating Reordering Techniques** ✓
- Community-based matrix reordering (RABBIT) is best overall
- RABBIT++: Transformations to improve RABBIT

OUTLINE

- Matrix Reordering Improves Performance ✓
- Methodology for Evaluating Reordering Techniques ✓
- Community-based matrix reordering (RABBIT) is best overall ⬅
- RABBIT++: Transformations to improve RABBIT

COMPARING EXISTING REORDERING TECHNIQUES

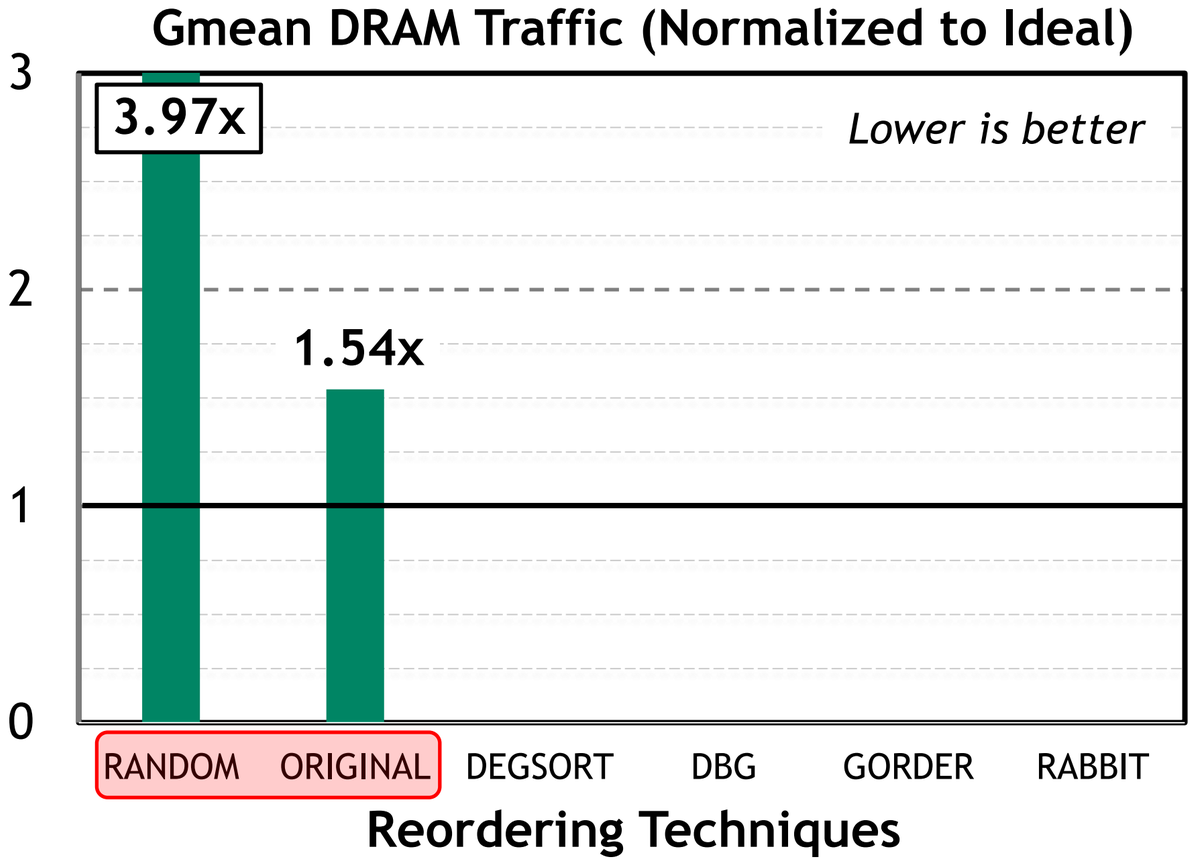
cuSPARSE SpMV on NVIDIA A6000 GPU



Ideal Traffic = Algorithmic Minimum DRAM Transfers

COMPARING EXISTING REORDERING TECHNIQUES

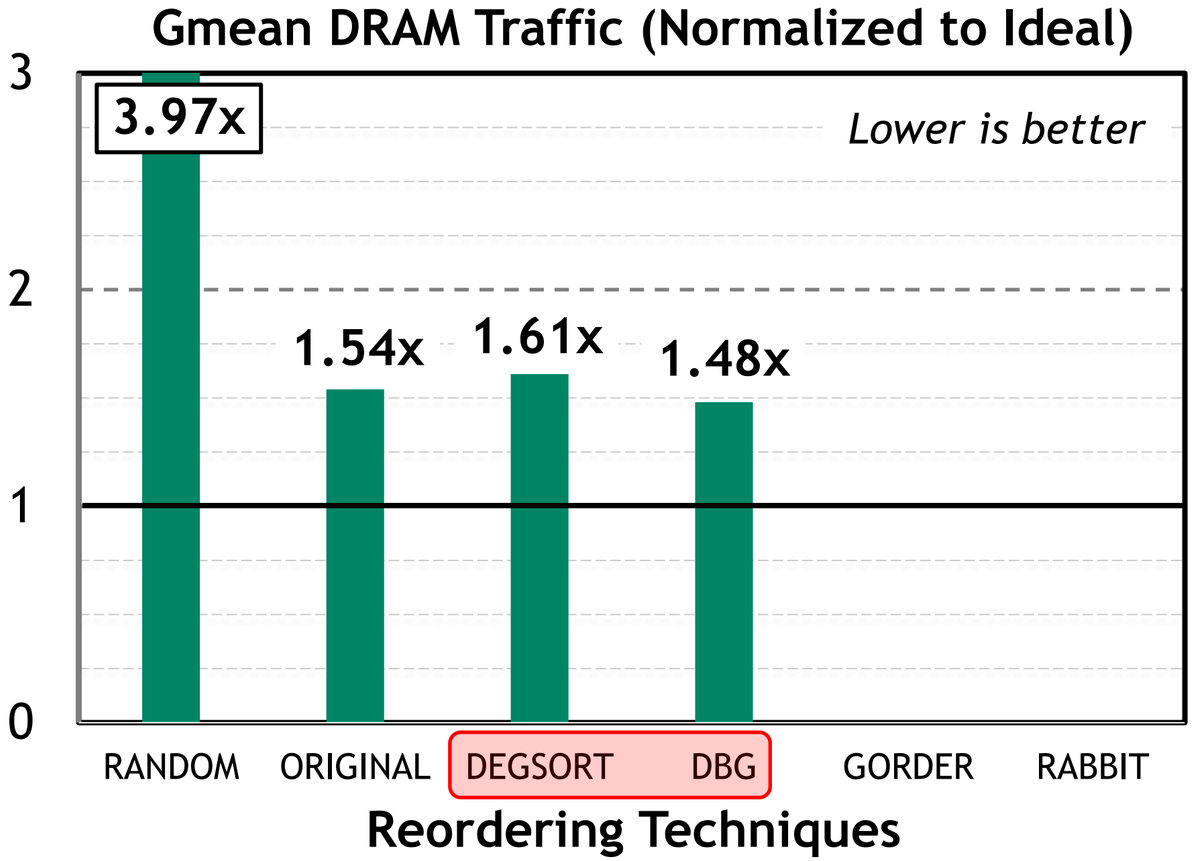
cuSPARSE SpMV on NVIDIA A6000 GPU



Ideal Traffic = Algorithmic Minimum DRAM Transfers

COMPARING EXISTING REORDERING TECHNIQUES

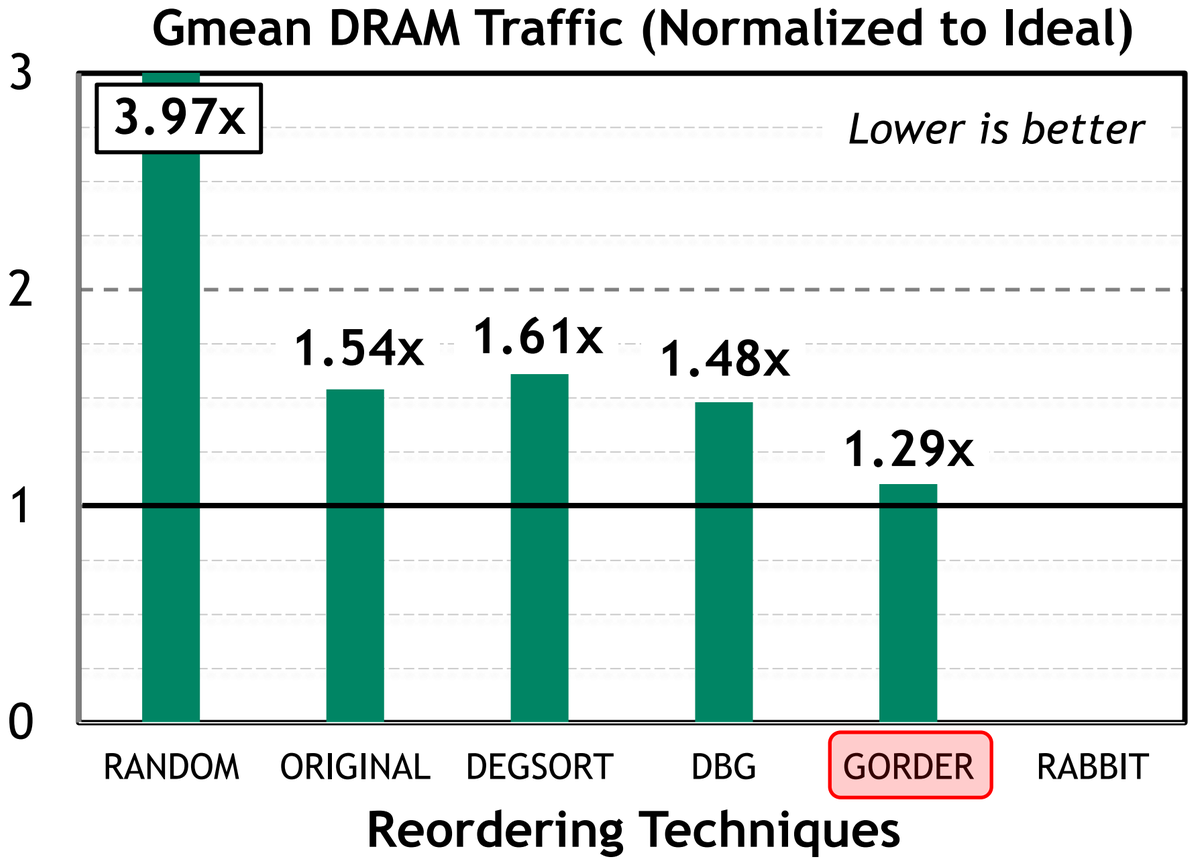
cuSPARSE SpMV on NVIDIA A6000 GPU



Ideal Traffic = Algorithmic Minimum DRAM Transfers

COMPARING EXISTING REORDERING TECHNIQUES

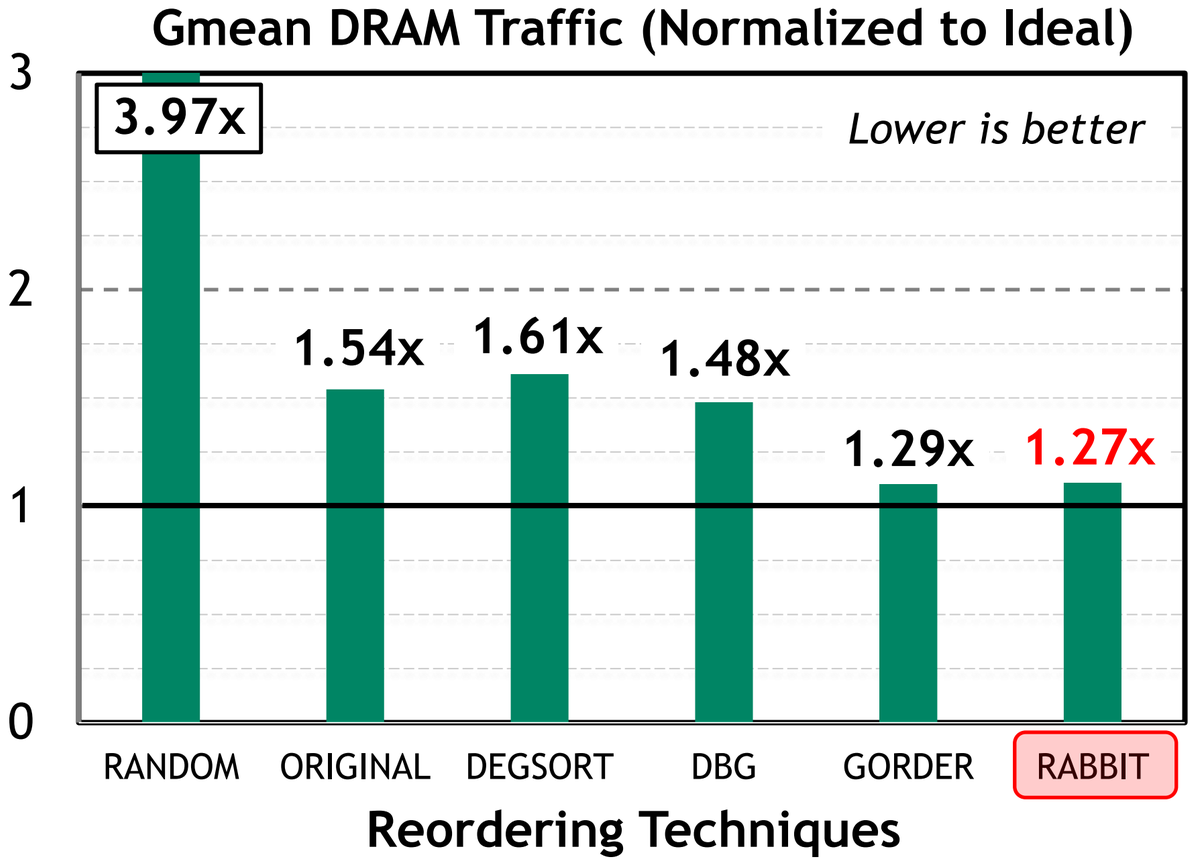
cuSPARSE SpMV on NVIDIA A6000 GPU

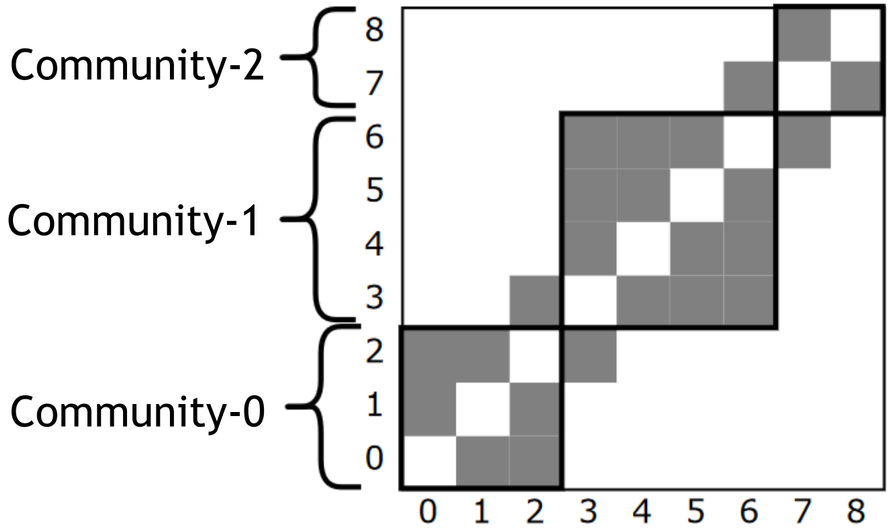


Ideal Traffic = Algorithmic Minimum DRAM Transfers

COMPARING EXISTING REORDERING TECHNIQUES

cuSPARSE SpMV on NVIDIA A6000 GPU

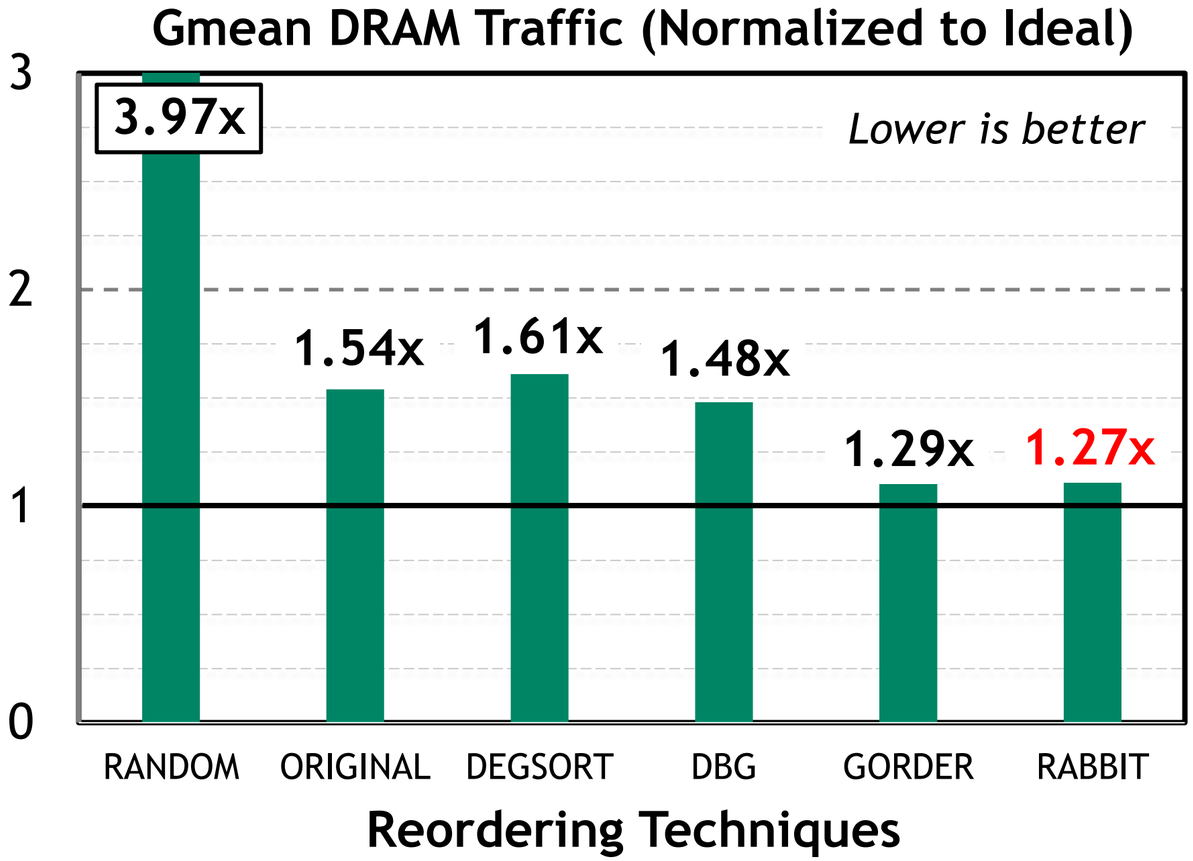


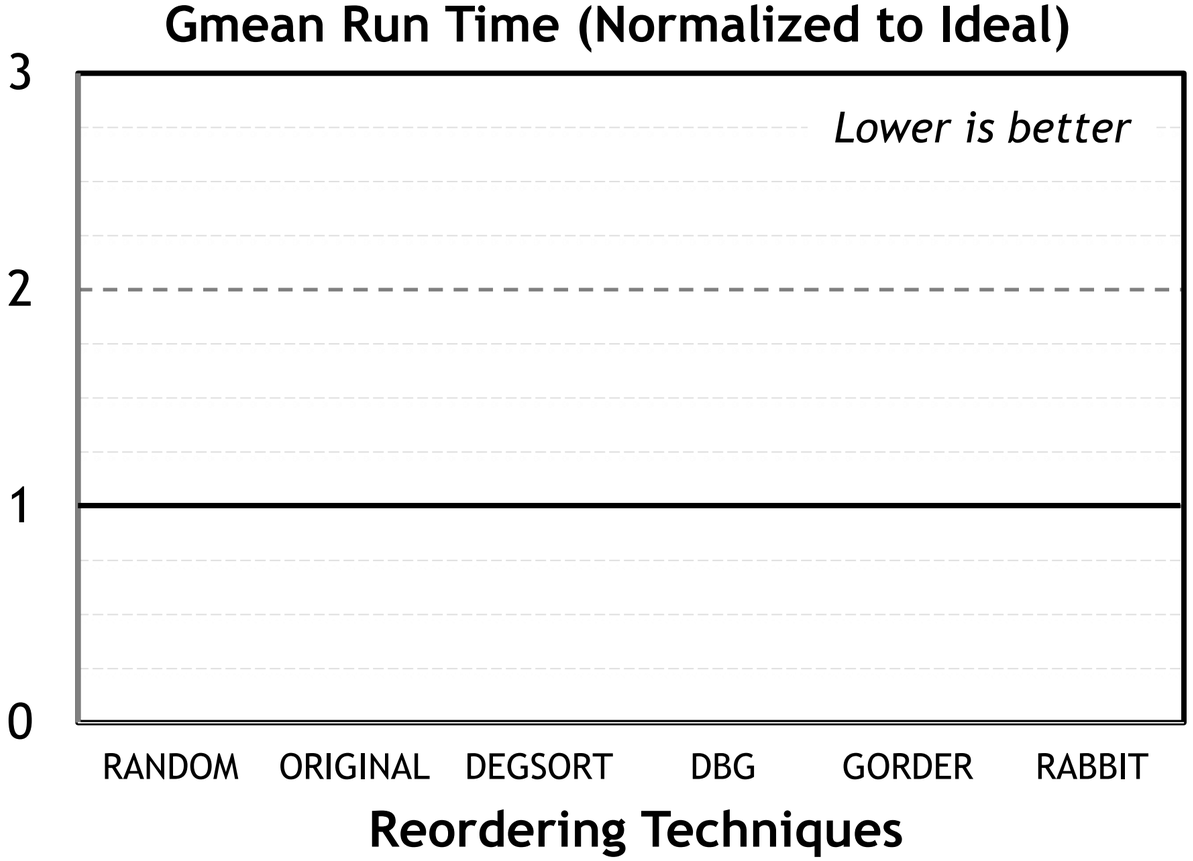


Ideal Traffic = Algorithmic Minimum DRAM Transfers

COMPARING EXISTING REORDERING TECHNIQUES

cuSPARSE SpMV on NVIDIA A6000 GPU



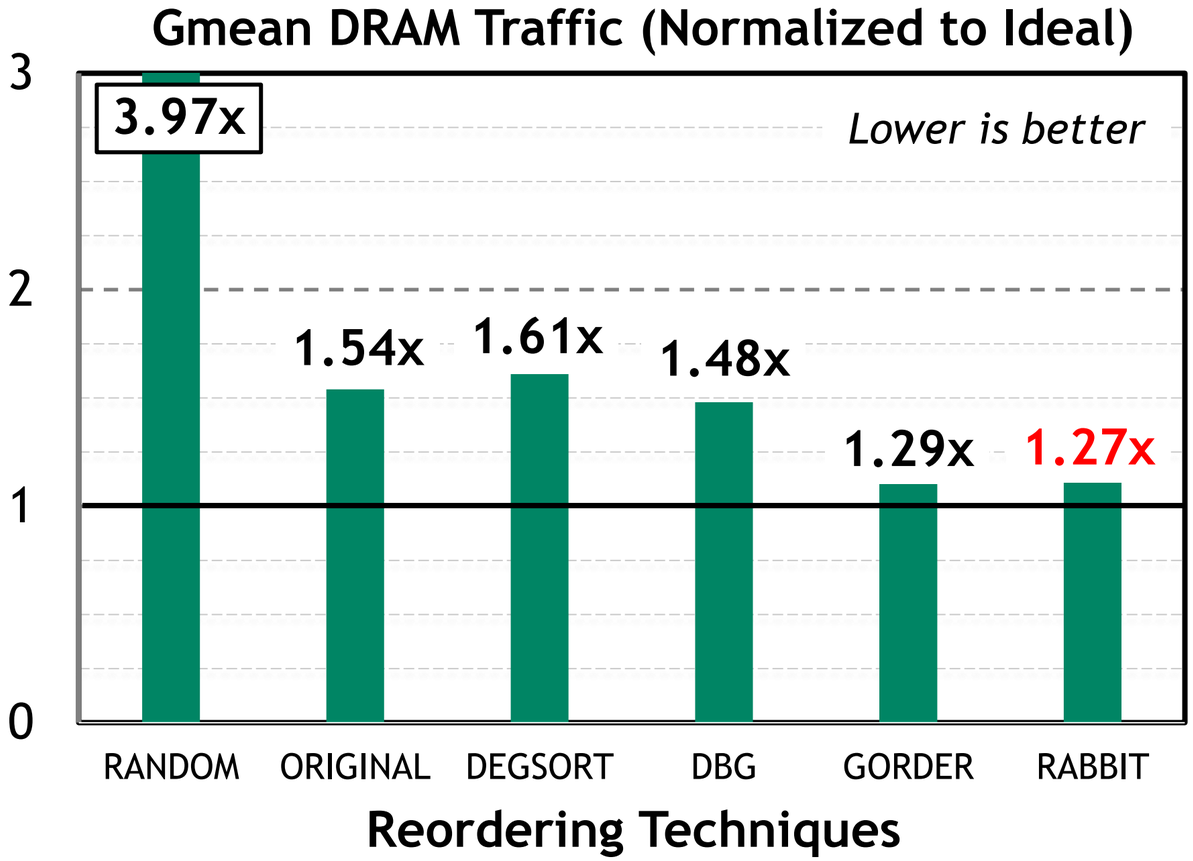


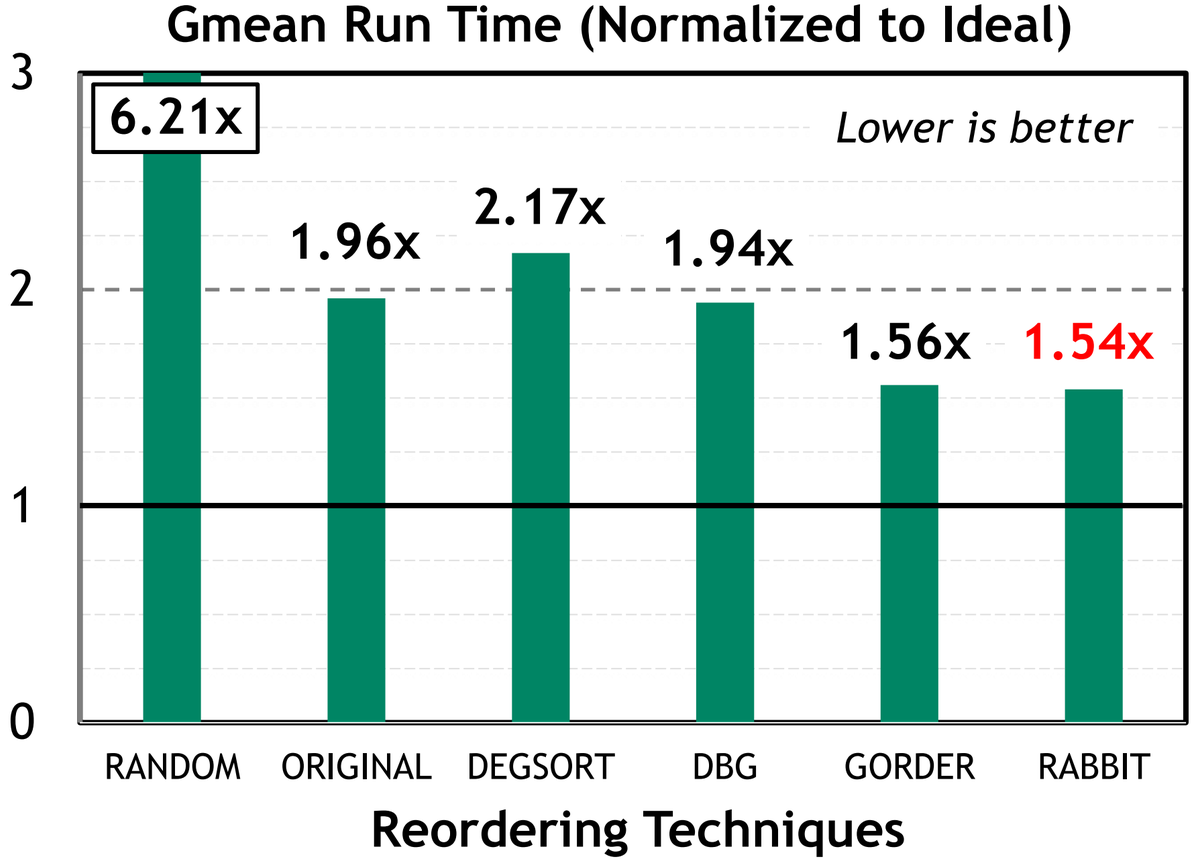
Ideal Traffic = Algorithmic Minimum DRAM Transfers

$$\text{Ideal Run Time} = \frac{\text{Algorithmic Min Traffic}}{\text{Peak DRAM Bandwidth}}$$

COMPARING EXISTING REORDERING TECHNIQUES

cuSPARSE SpMV on NVIDIA A6000 GPU





Ideal Traffic = Algorithmic Minimum DRAM Transfers

$$\text{Ideal Run Time} = \frac{\text{Algorithmic Min Traffic}}{\text{Peak DRAM Bandwidth}}$$

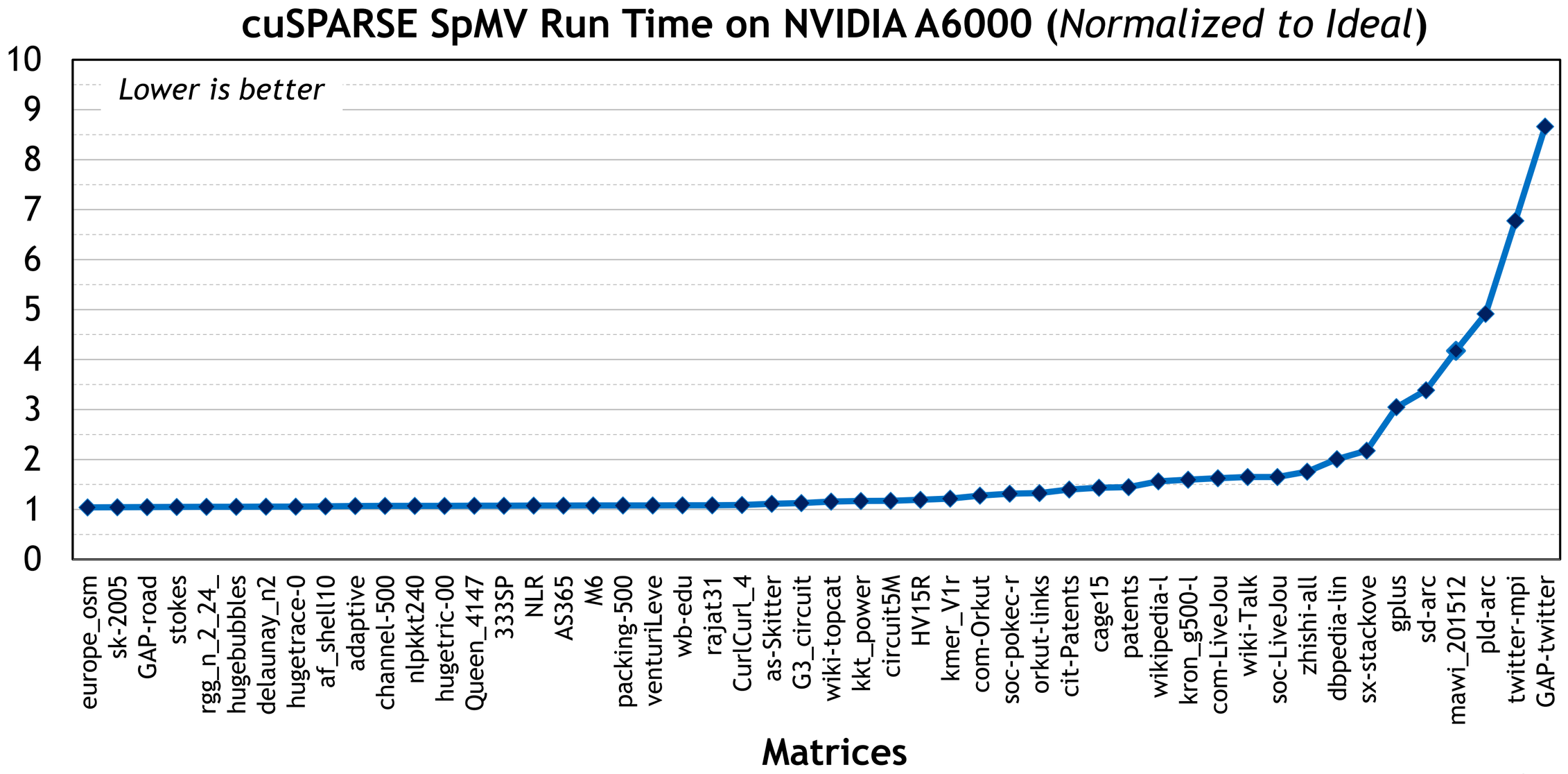
OUTLINE

- **Matrix Reordering Improves Performance ✓**
- **Methodology for Evaluating Reordering Techniques ✓**
- **Community-based matrix reordering (RABBIT) is best overall ✓**
- RABBIT++: Transformations to improve RABBIT

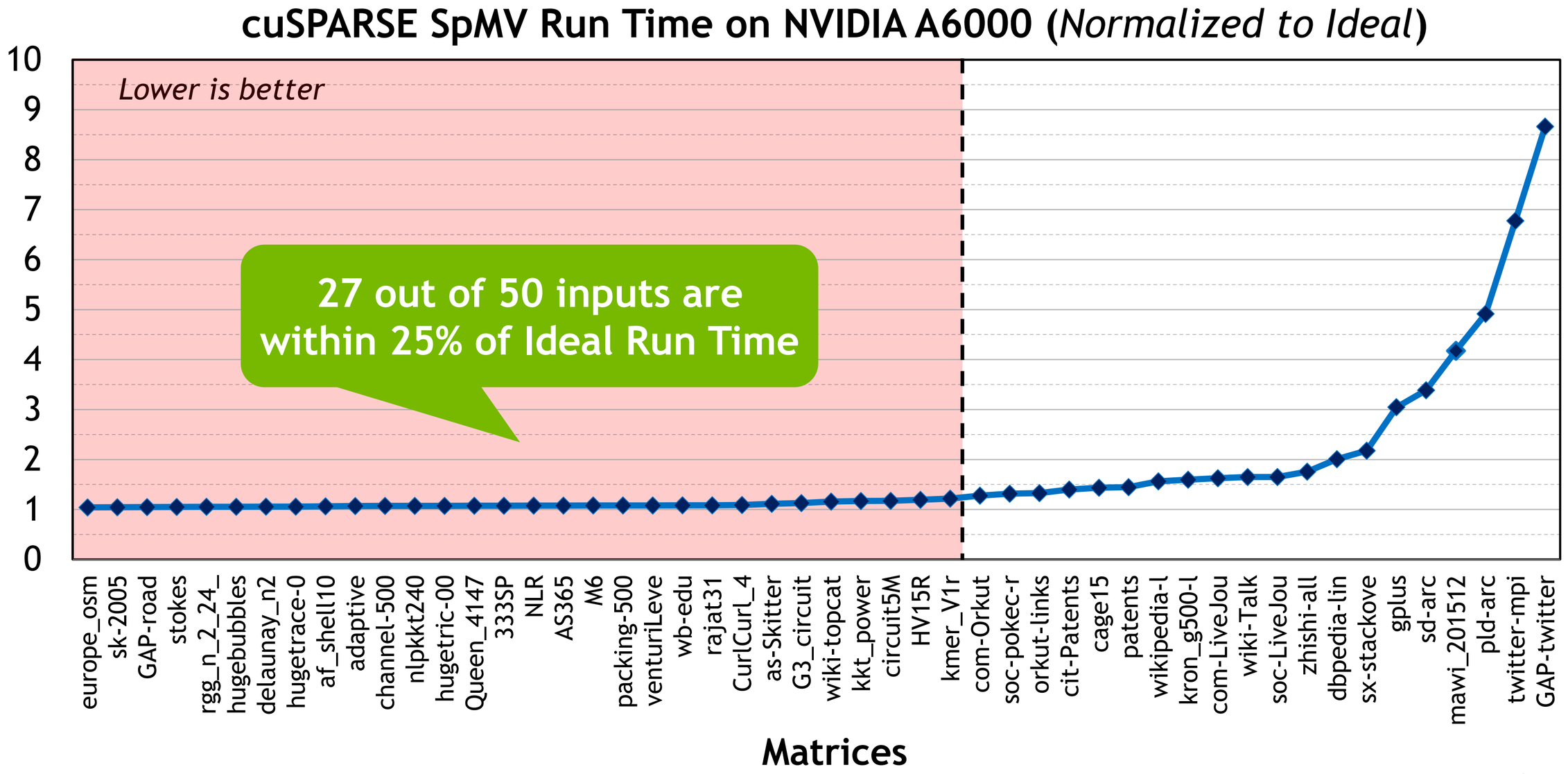
OUTLINE

- Matrix Reordering Improves Performance ✓
- Methodology for Evaluating Reordering Techniques ✓
- Community-based matrix reordering (RABBIT) is best overall ✓
- RABBIT++: Transformations to improve RABBIT ⬅

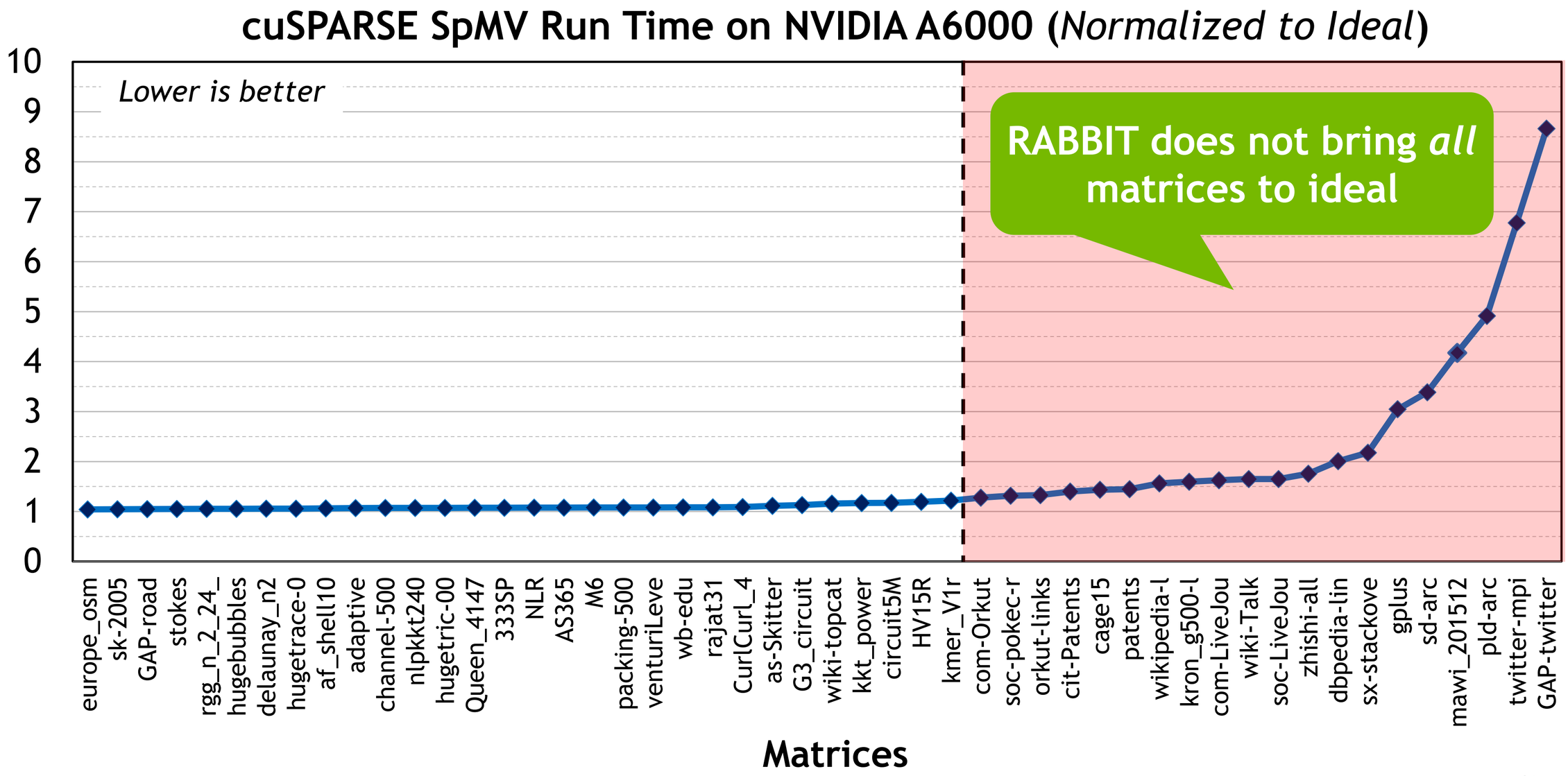
ZOOMING IN ON RABBIT ORDERING



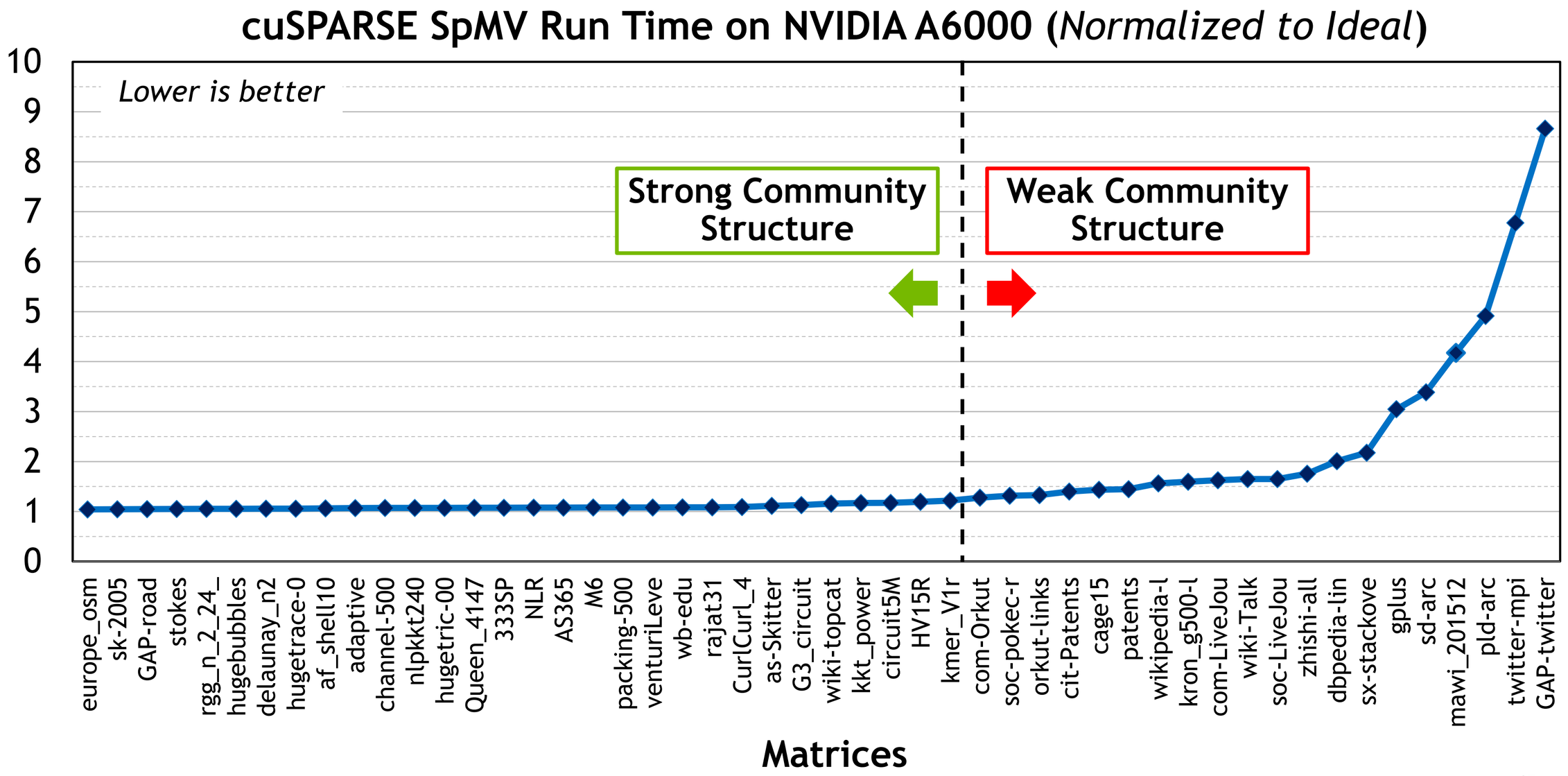
ZOOMING IN ON RABBIT ORDERING



ZOOMING IN ON RABBIT ORDERING

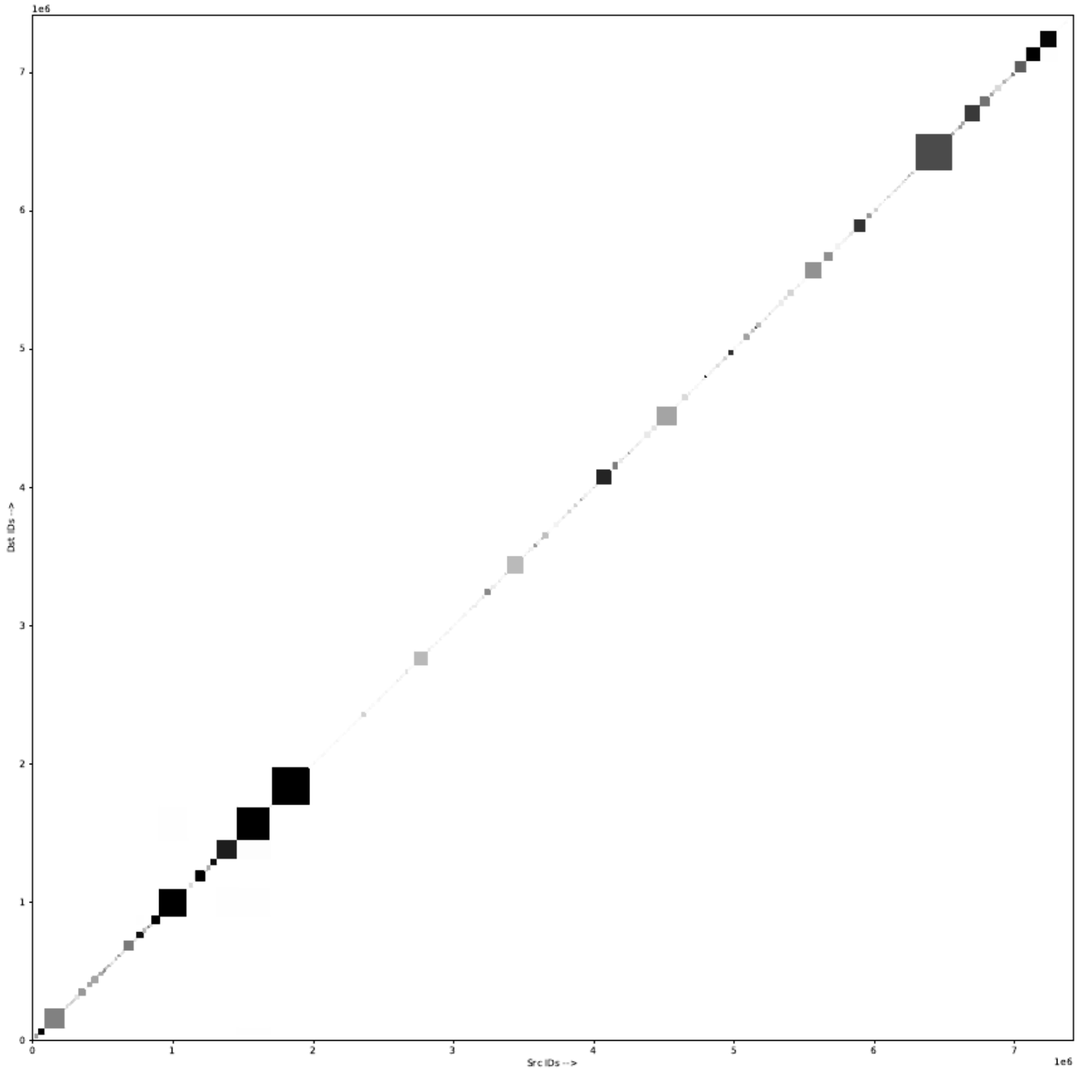


ZOOMING IN ON RABBIT ORDERING



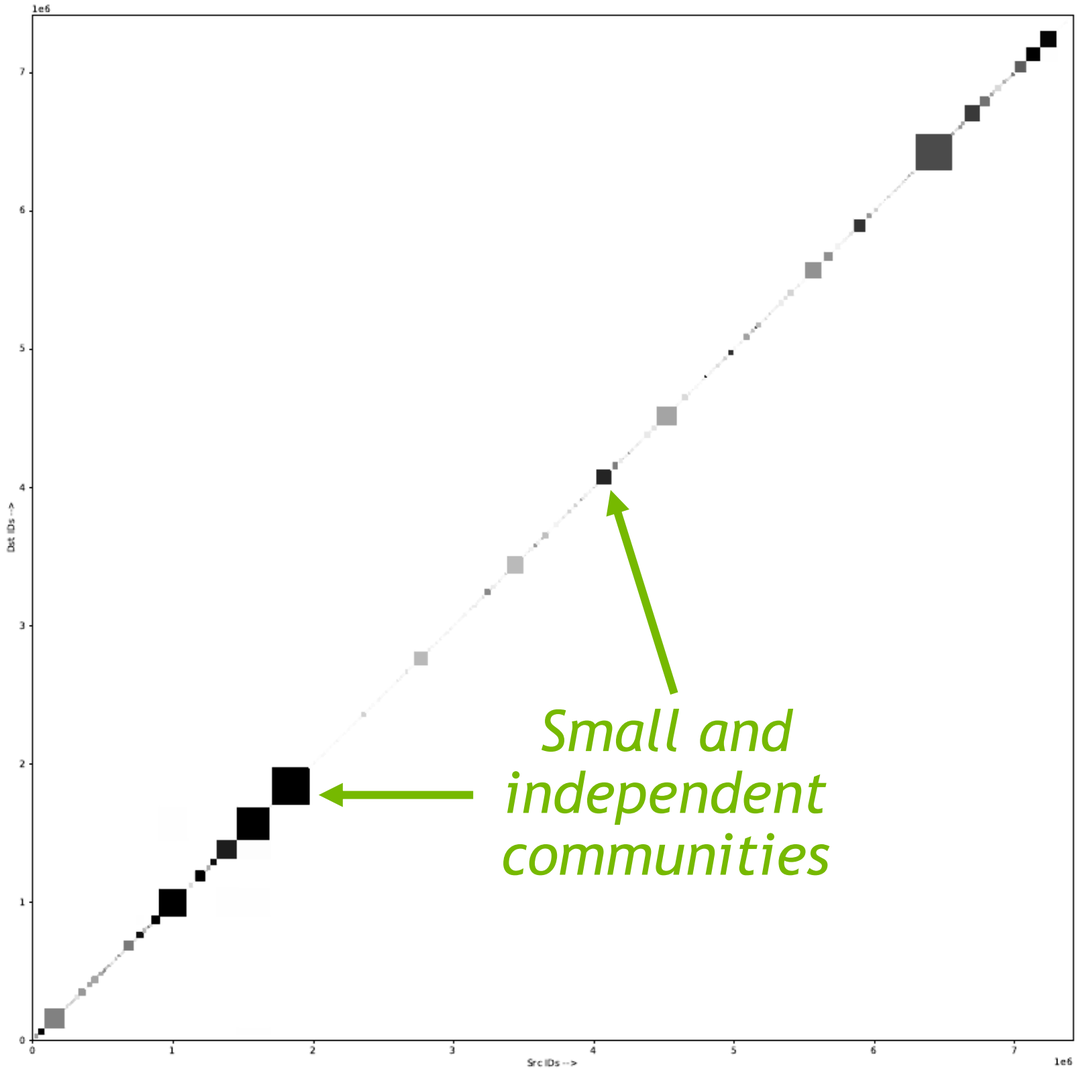
LIMITATIONS OF RABBIT

Strong Community Structure



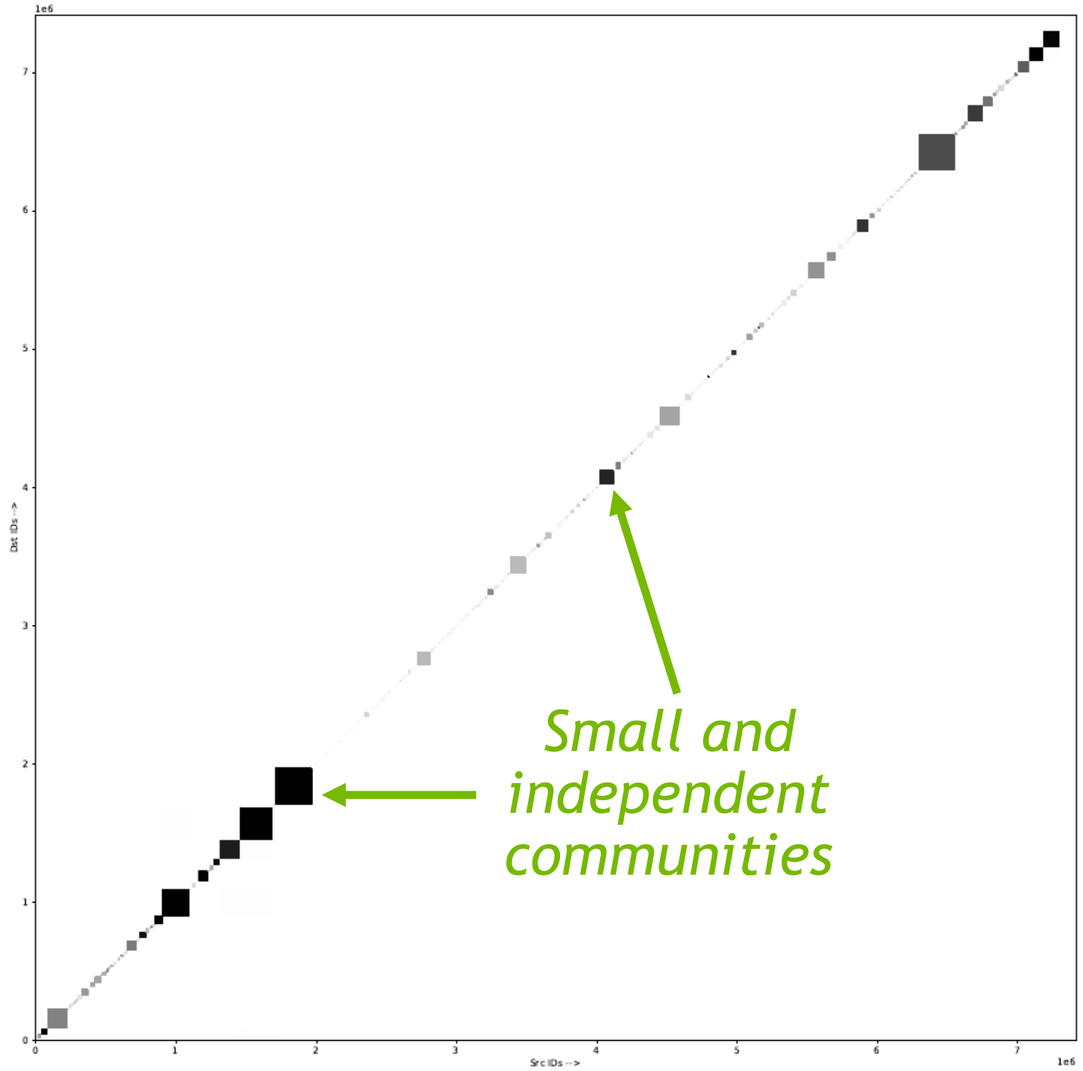
LIMITATIONS OF RABBIT

Strong Community Structure

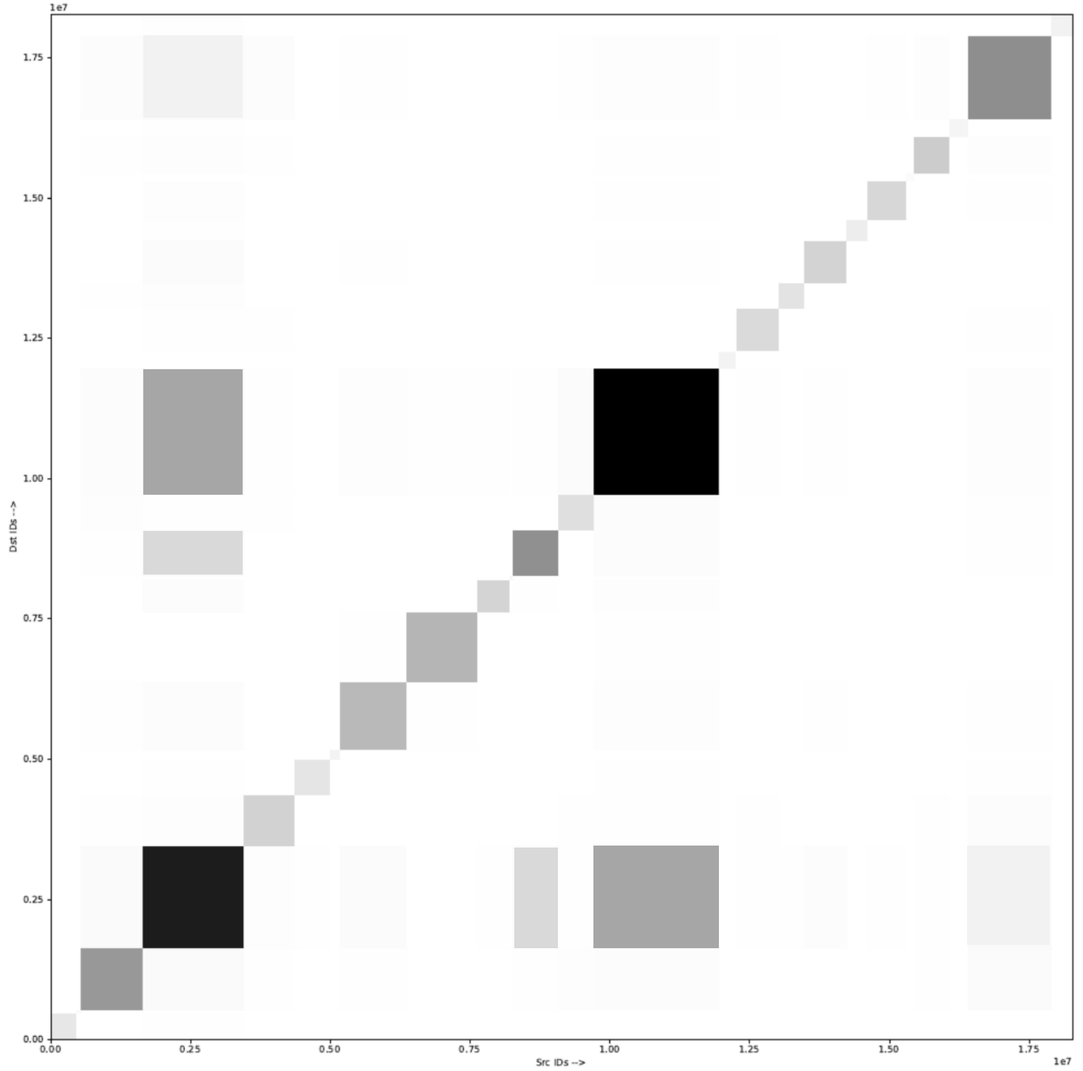


LIMITATIONS OF RABBIT

Strong Community Structure

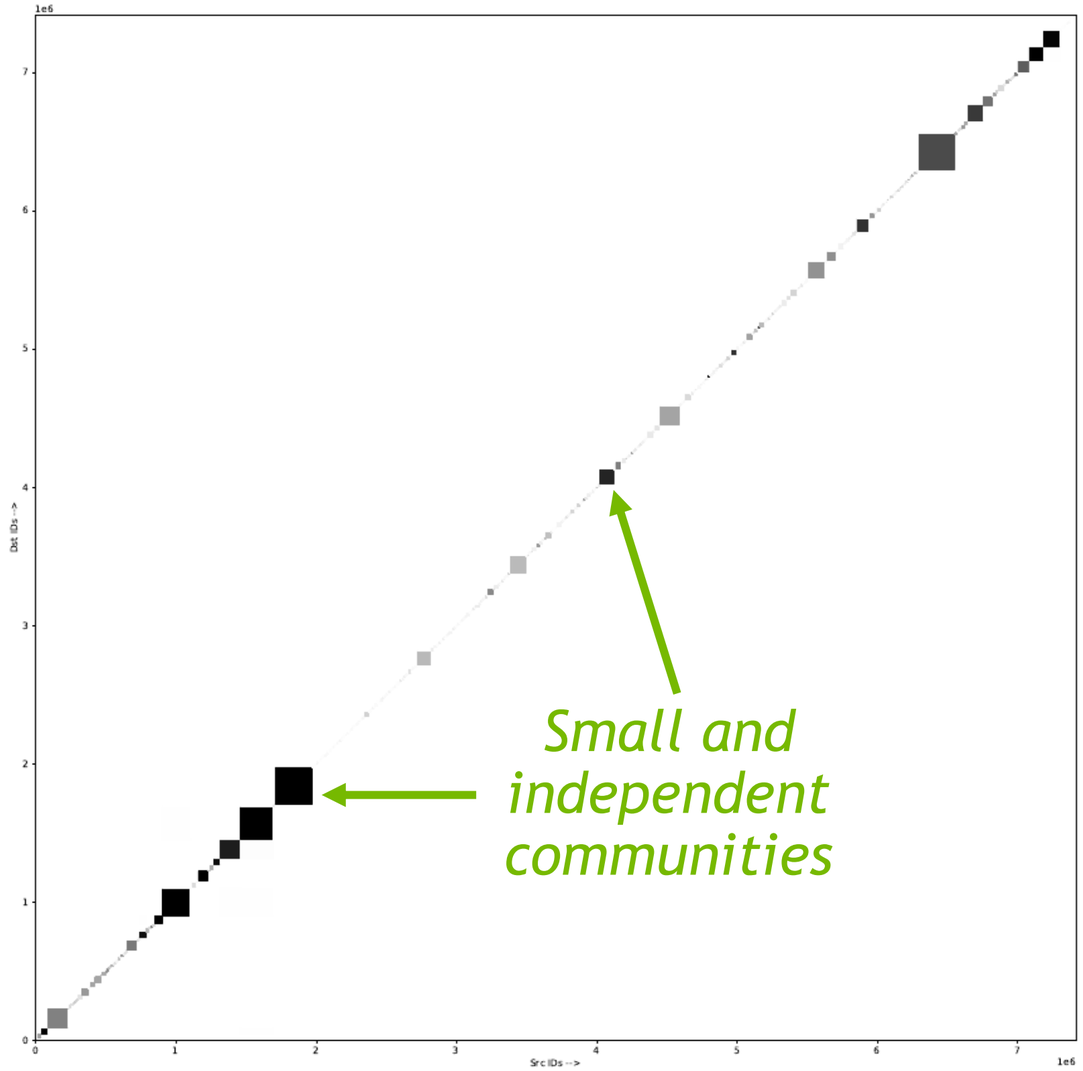


Weak Community Structure

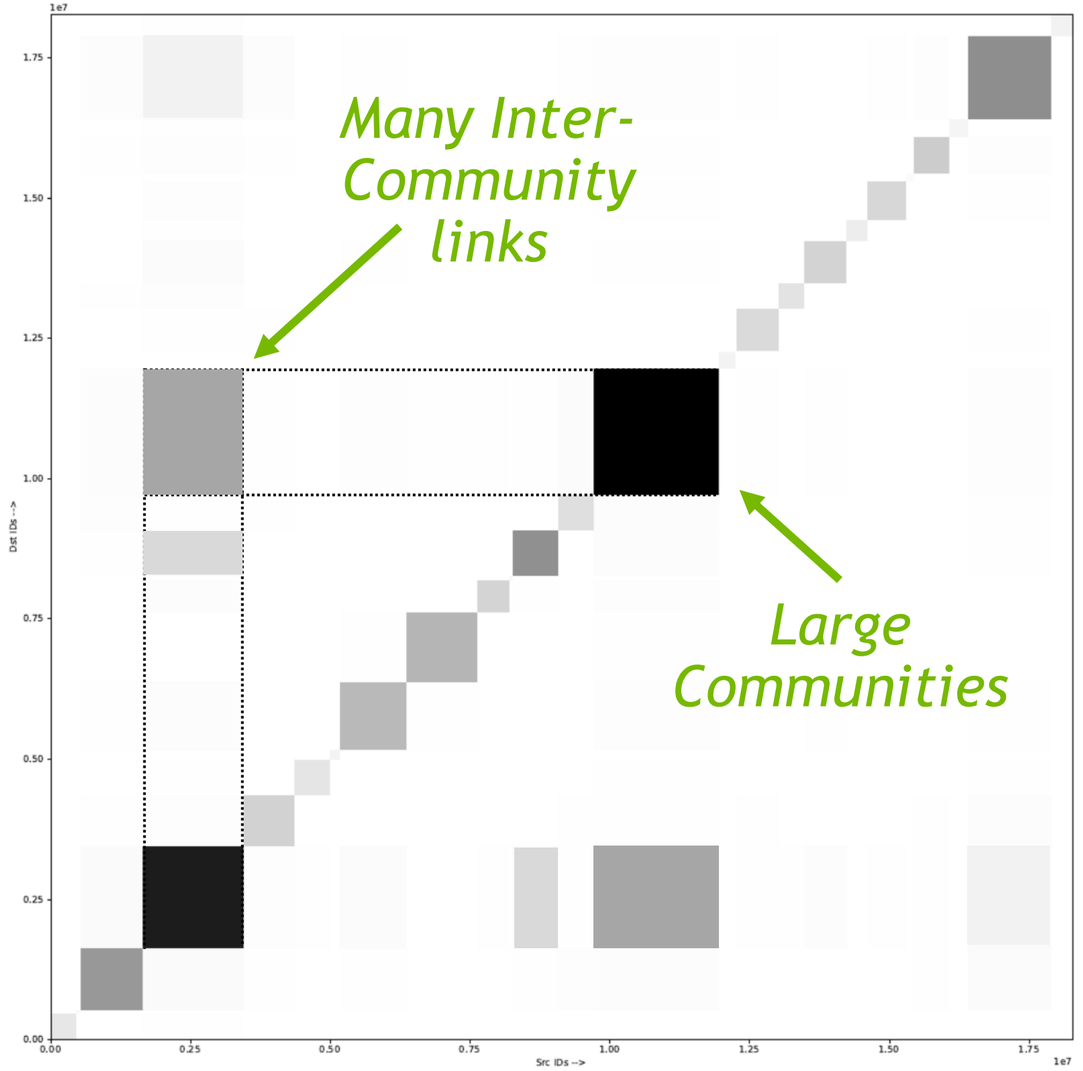


LIMITATIONS OF RABBIT

Strong Community Structure

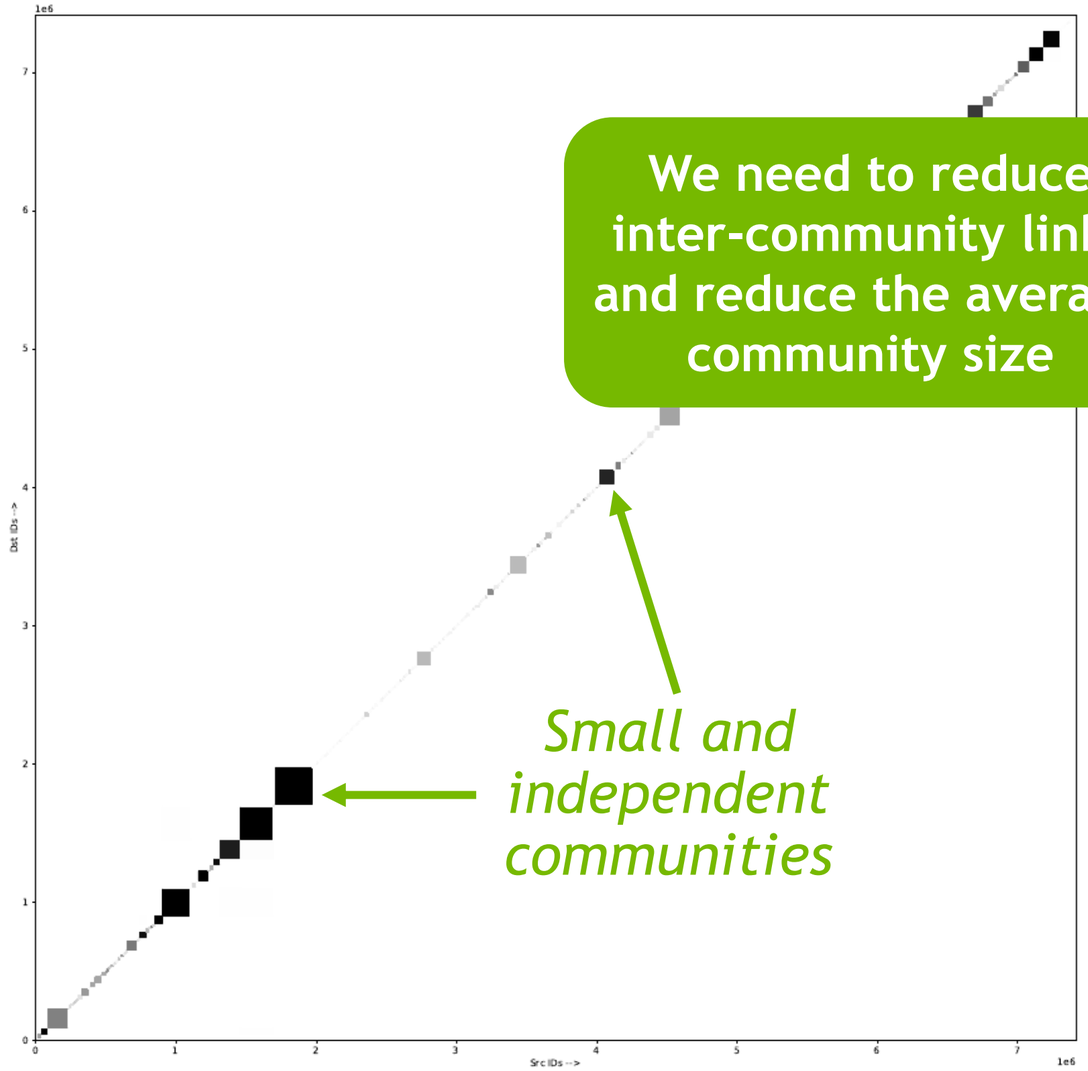


Weak Community Structure



LIMITATIONS OF RABBIT

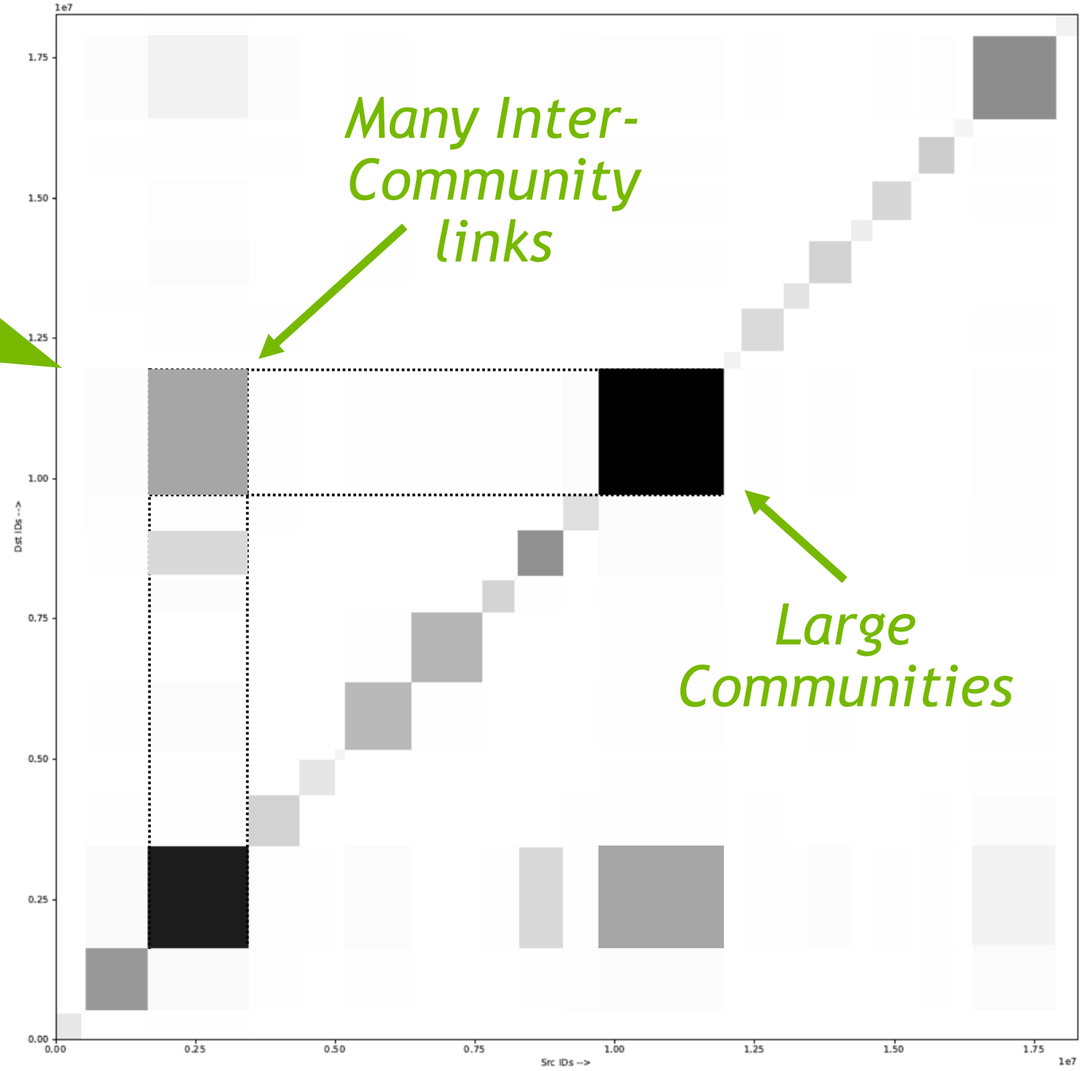
Strong Community Structure



We need to reduce inter-community links and reduce the average community size

Small and independent communities

Weak Community Structure

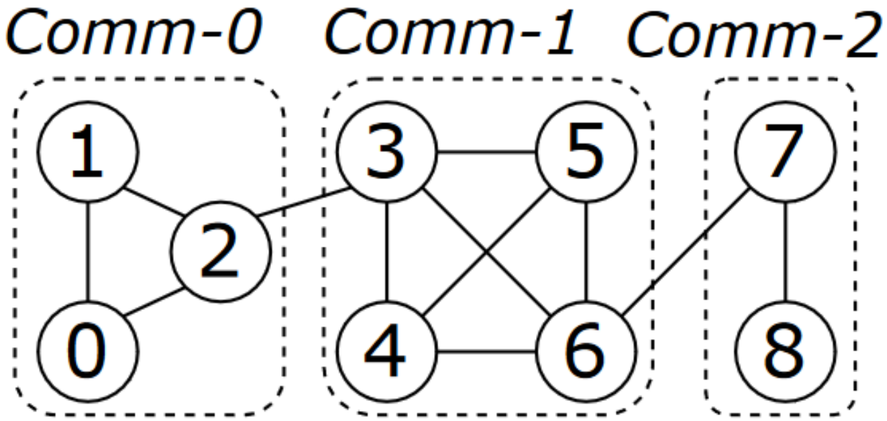


Many Inter-Community links

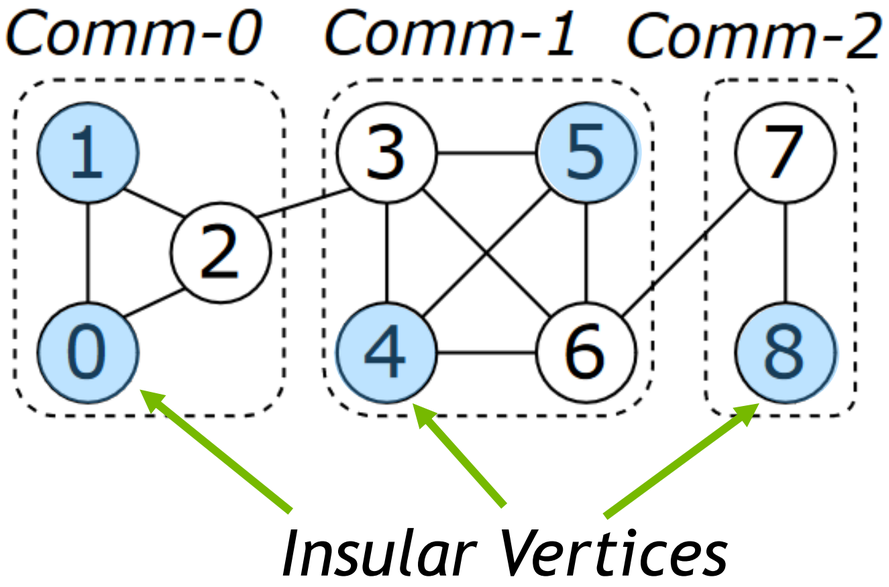
Large Communities

NVIDIA.

PROPERTIES OF INPUTS WITH WEAK COMMUNITY STRUCTURE



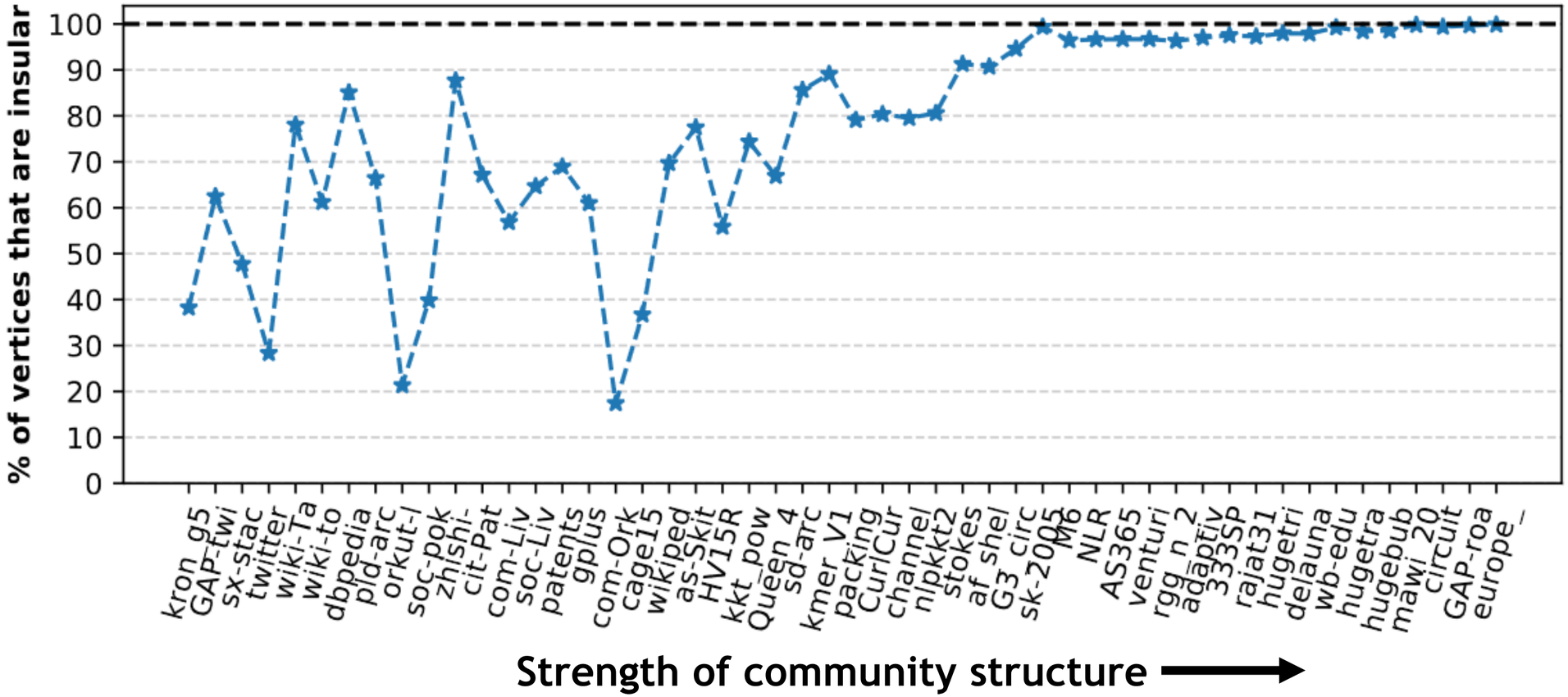
PROPERTIES OF INPUTS WITH WEAK COMMUNITY STRUCTURE



Insular Vertex:
All neighbors are in the same community

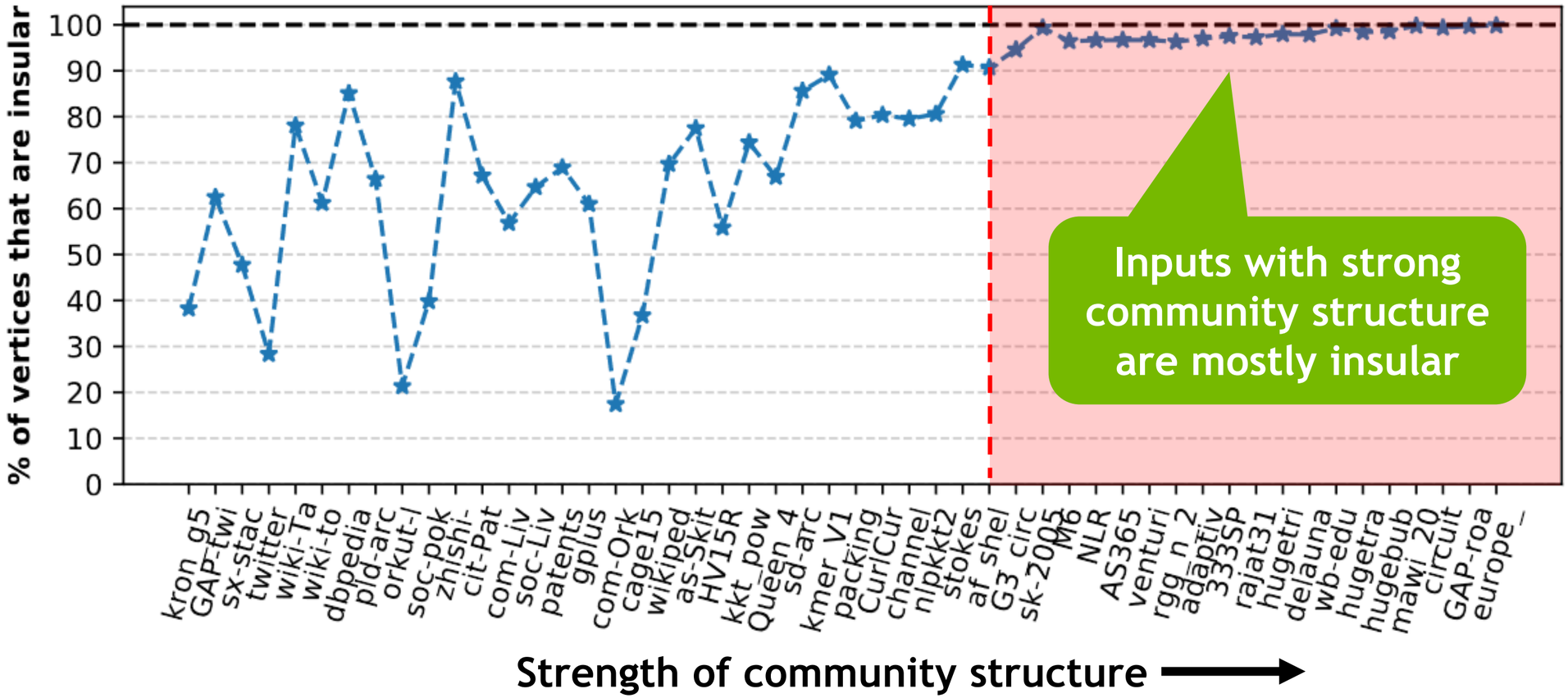
PROPERTIES OF INPUTS WITH WEAK COMMUNITY STRUCTURE

- **PROPERTY #1:** A large percentage of the matrix is ***insular***



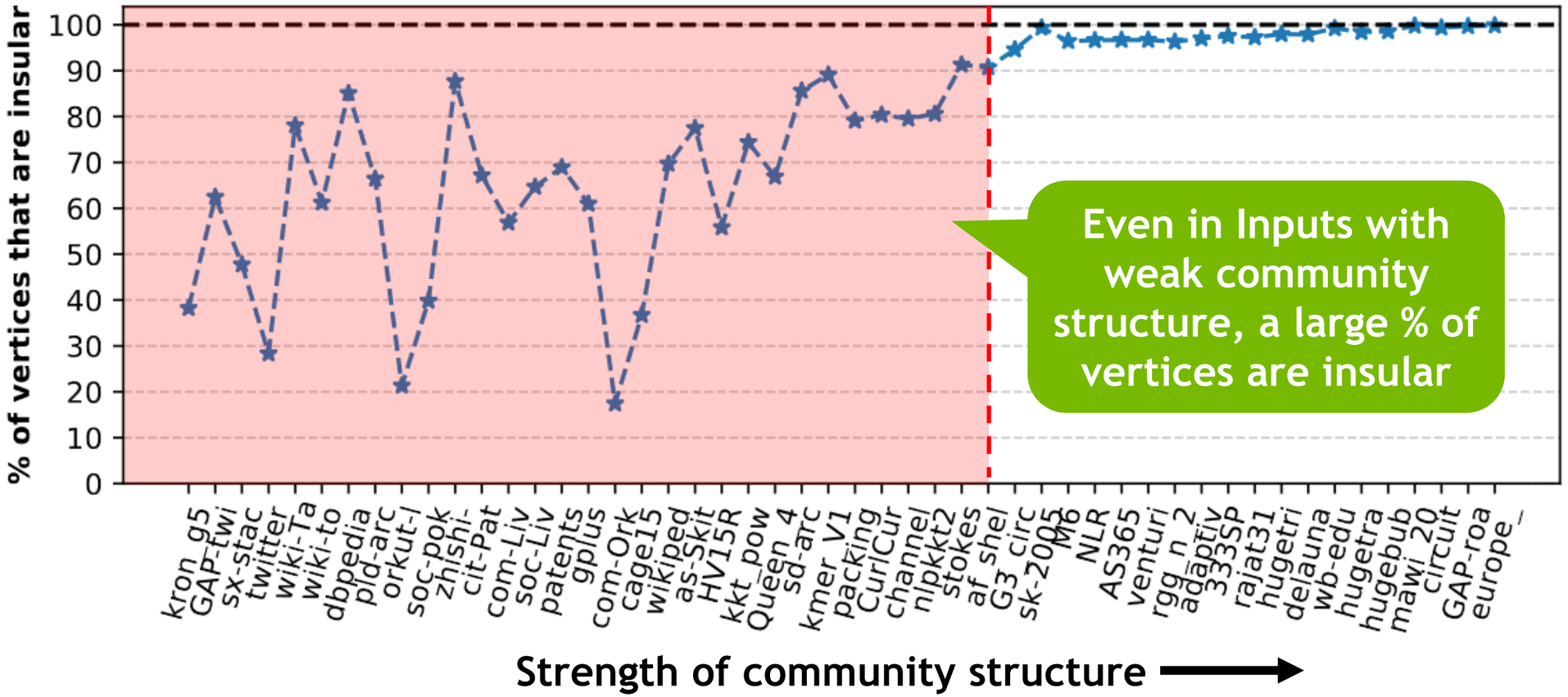
PROPERTIES OF INPUTS WITH WEAK COMMUNITY STRUCTURE

- **PROPERTY #1:** A large percentage of the matrix is ***insular***



PROPERTIES OF INPUTS WITH WEAK COMMUNITY STRUCTURE

- **PROPERTY #1:** A large percentage of the matrix is ***insular***

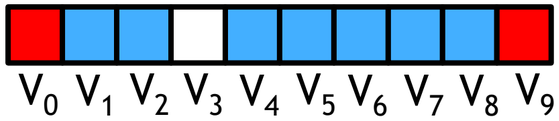


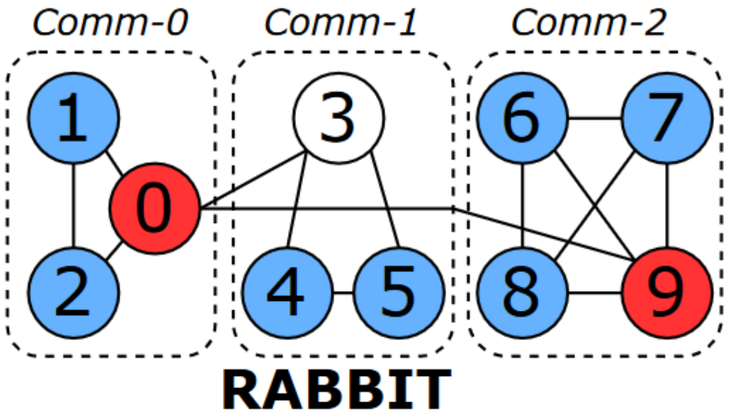
PROPERTIES OF INPUTS WITH WEAK COMMUNITY STRUCTURE

- **PROPERTY #1:** A large percentage of the matrix is ***insular***

- **PROPERTY #2:** ***Hubs*** account for most of the inter-community links
  - *Highly-connected hubs (vertices with degree > average degree) are responsible for 86% for all inter-community links*

ENHANCING RABBIT

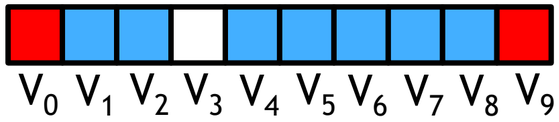


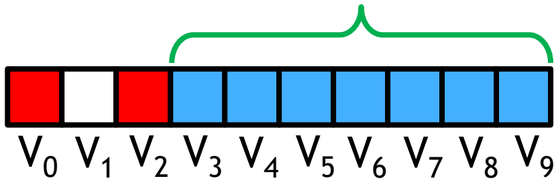


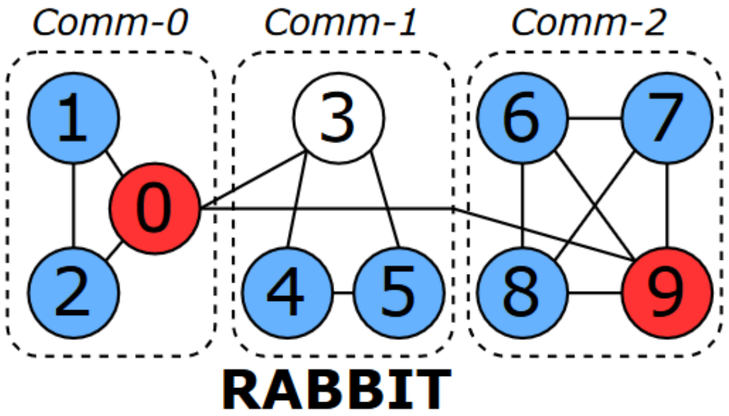
Insular Vertices

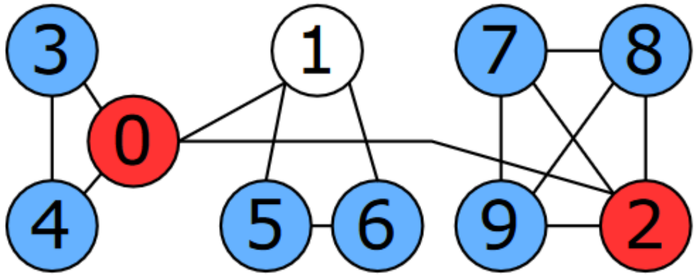
Hub Vertices

ENHANCING RABBIT







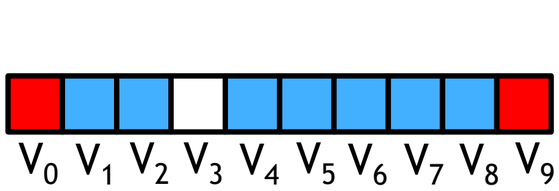


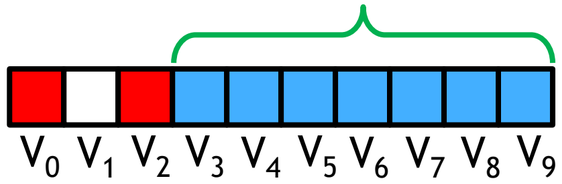
Insular Vertices

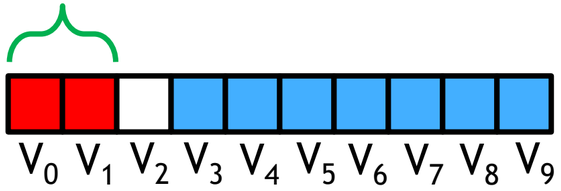
Hub Vertices

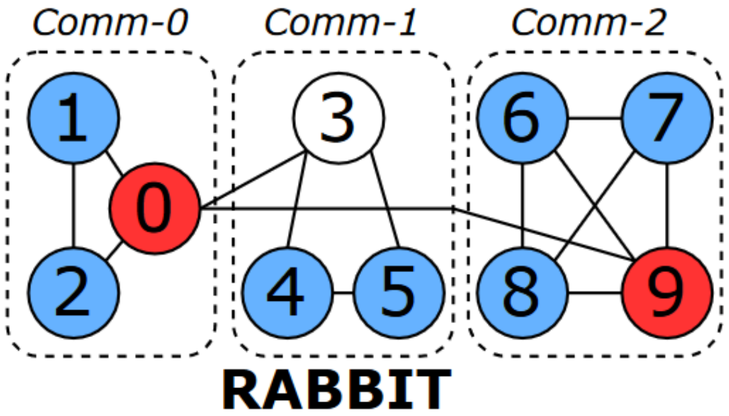
Transformation #1:
Group Insular Vertices

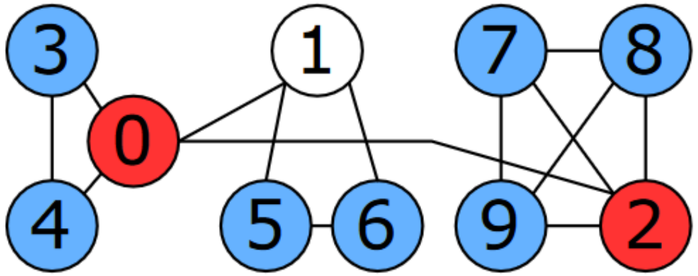
ENHANCING RABBIT

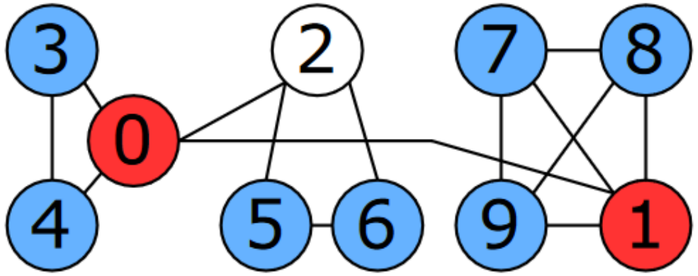












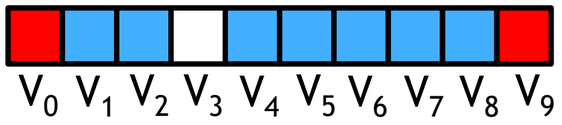
Insular Vertices

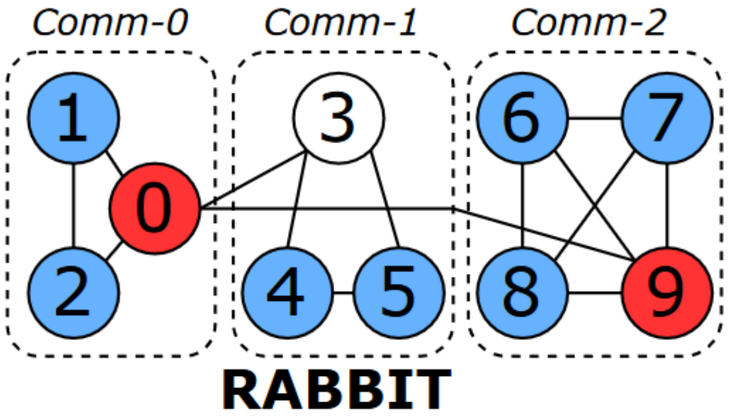
Hub Vertices

Transformation #1:
Group Insular Vertices

Transformation #2:
Group Hub Vertices

ENHANCING RABBIT

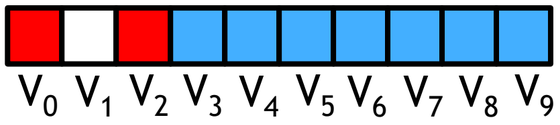


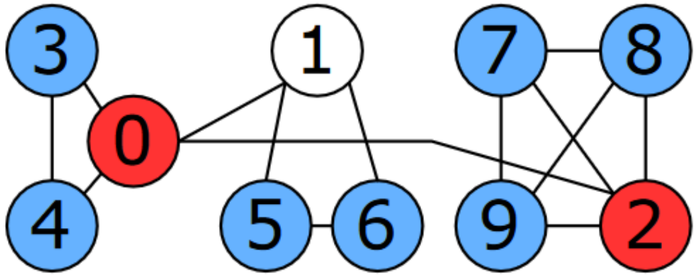


RABBIT

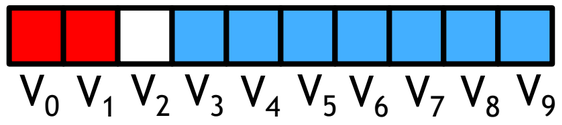
Insular Vertices

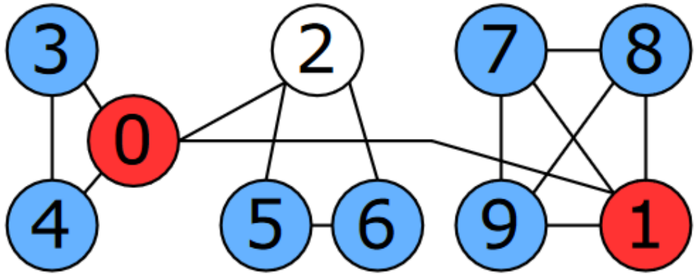
Hub Vertices





Transformation #1:
Group Insular Vertices



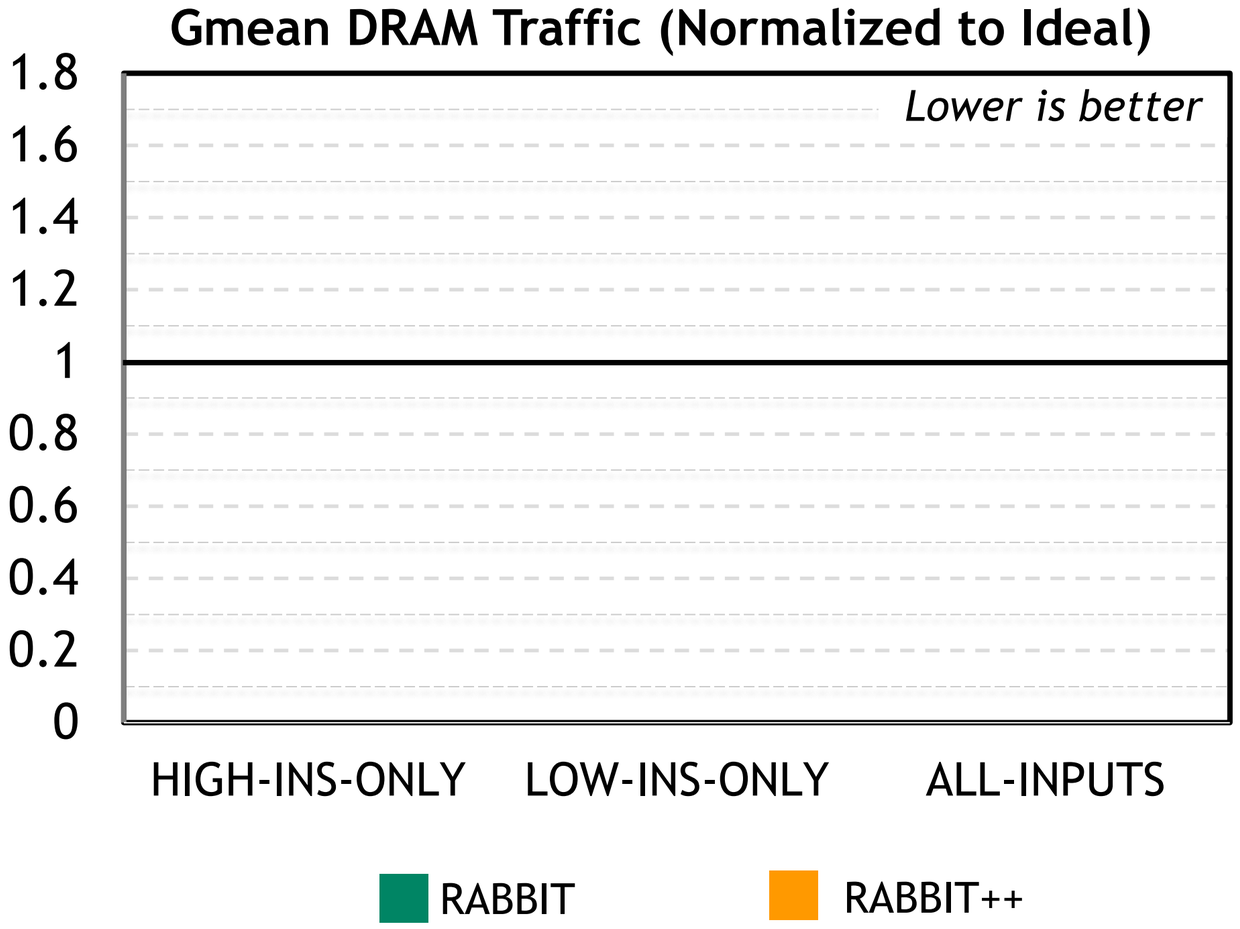


Transformation #2:
Group Hub Vertices

RABBIT++

PERFORMANCE IMPROVEMENTS WITH RABBIT++

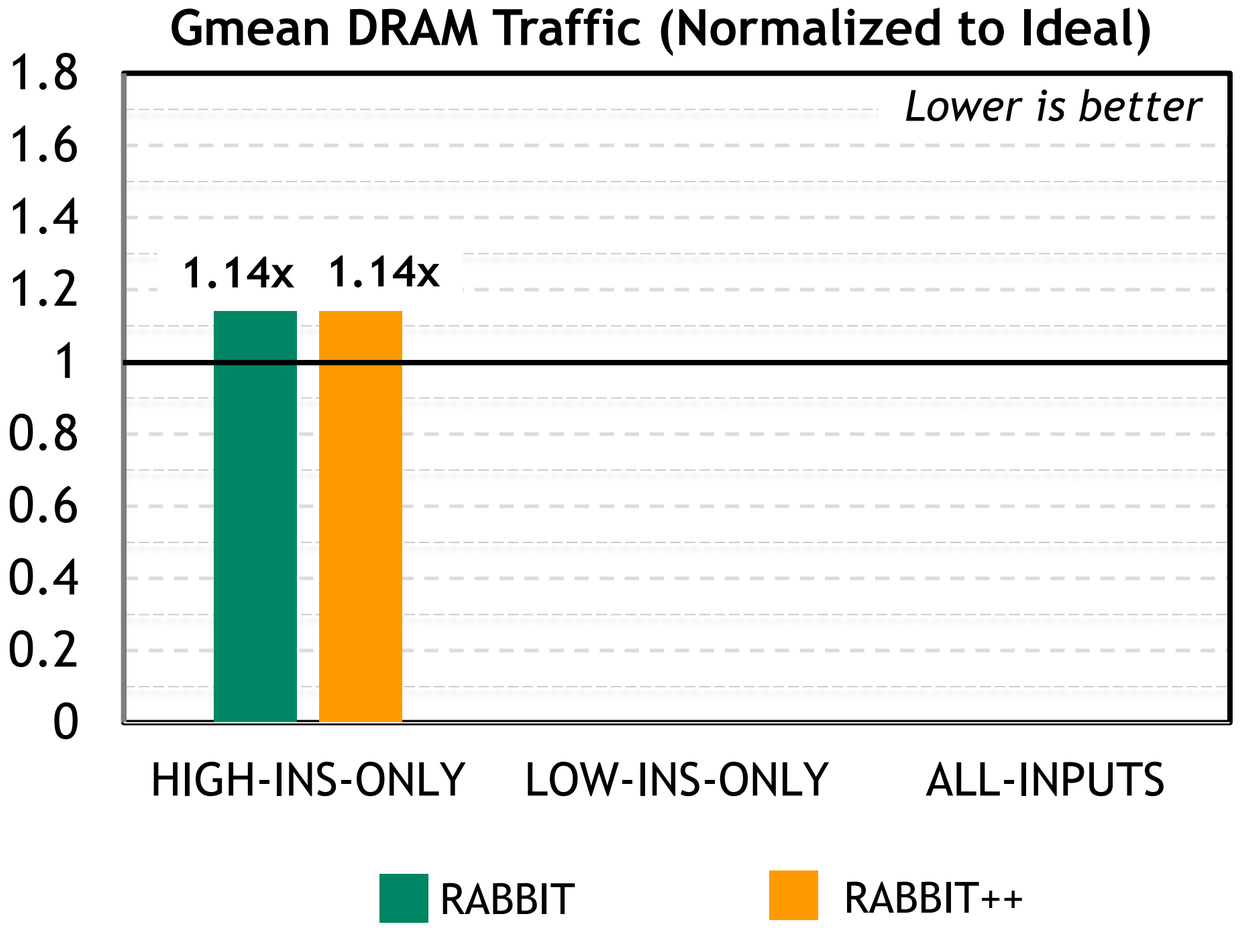
cuSPARSE SpMV on NVIDIA A6000 GPU



Ideal Traffic = Algorithmic Minimum DRAM Transfers

PERFORMANCE IMPROVEMENTS WITH RABBIT++

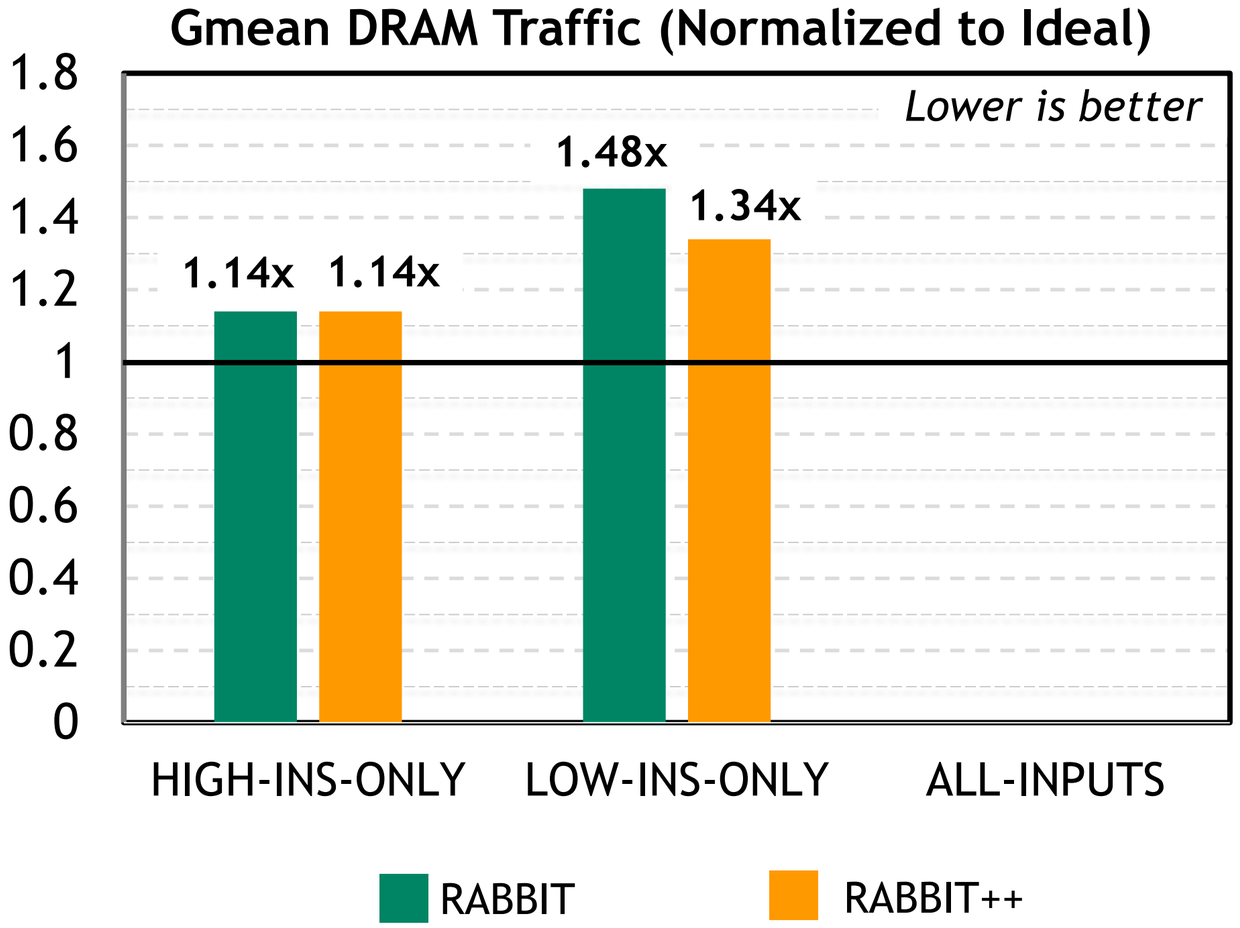
cuSPARSE SpMV on NVIDIA A6000 GPU



Ideal Traffic = Algorithmic Minimum DRAM Transfers

PERFORMANCE IMPROVEMENTS WITH RABBIT++

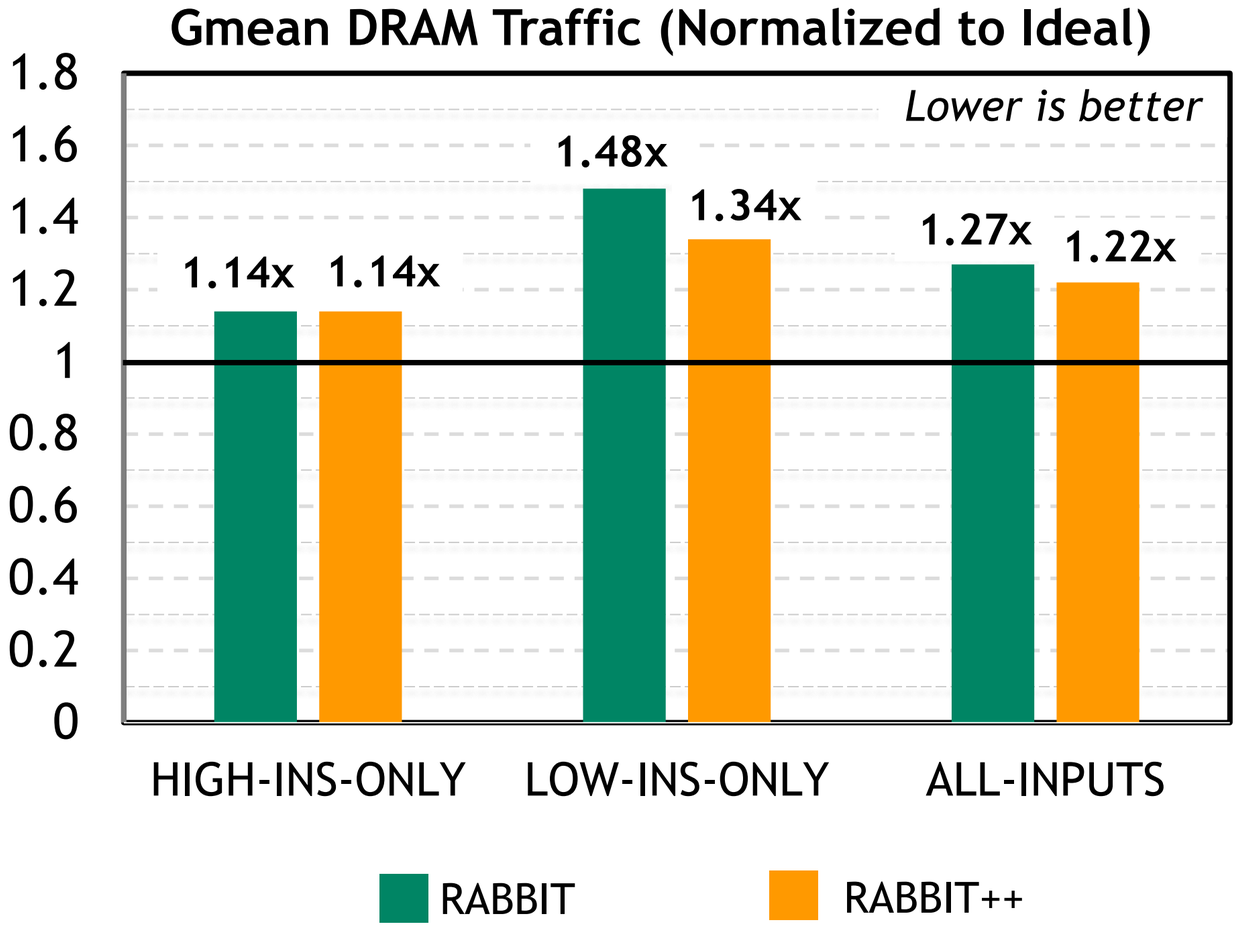
cuSPARSE SpMV on NVIDIA A6000 GPU



Ideal Traffic = Algorithmic Minimum DRAM Transfers

PERFORMANCE IMPROVEMENTS WITH RABBIT++

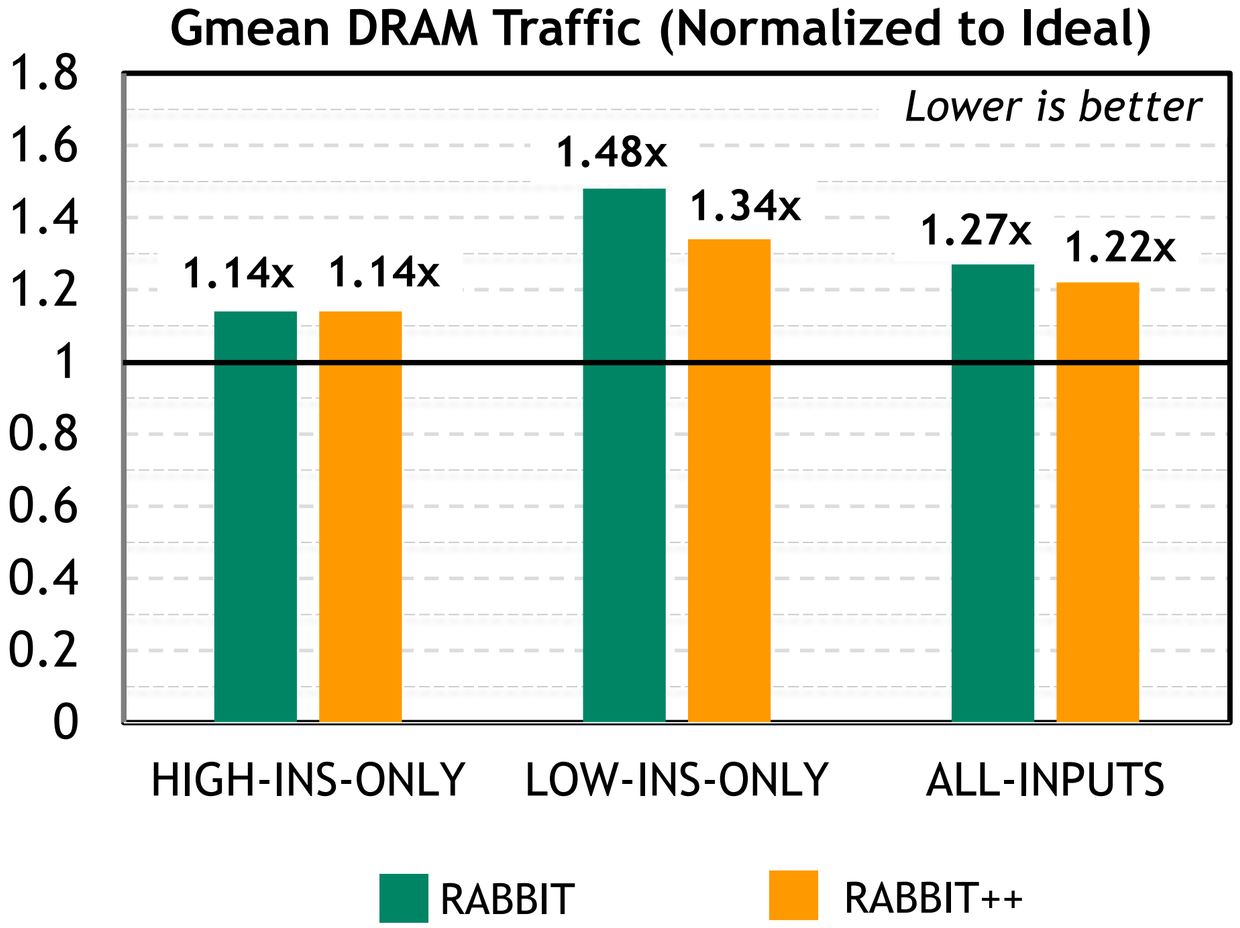
cuSPARSE SpMV on NVIDIA A6000 GPU

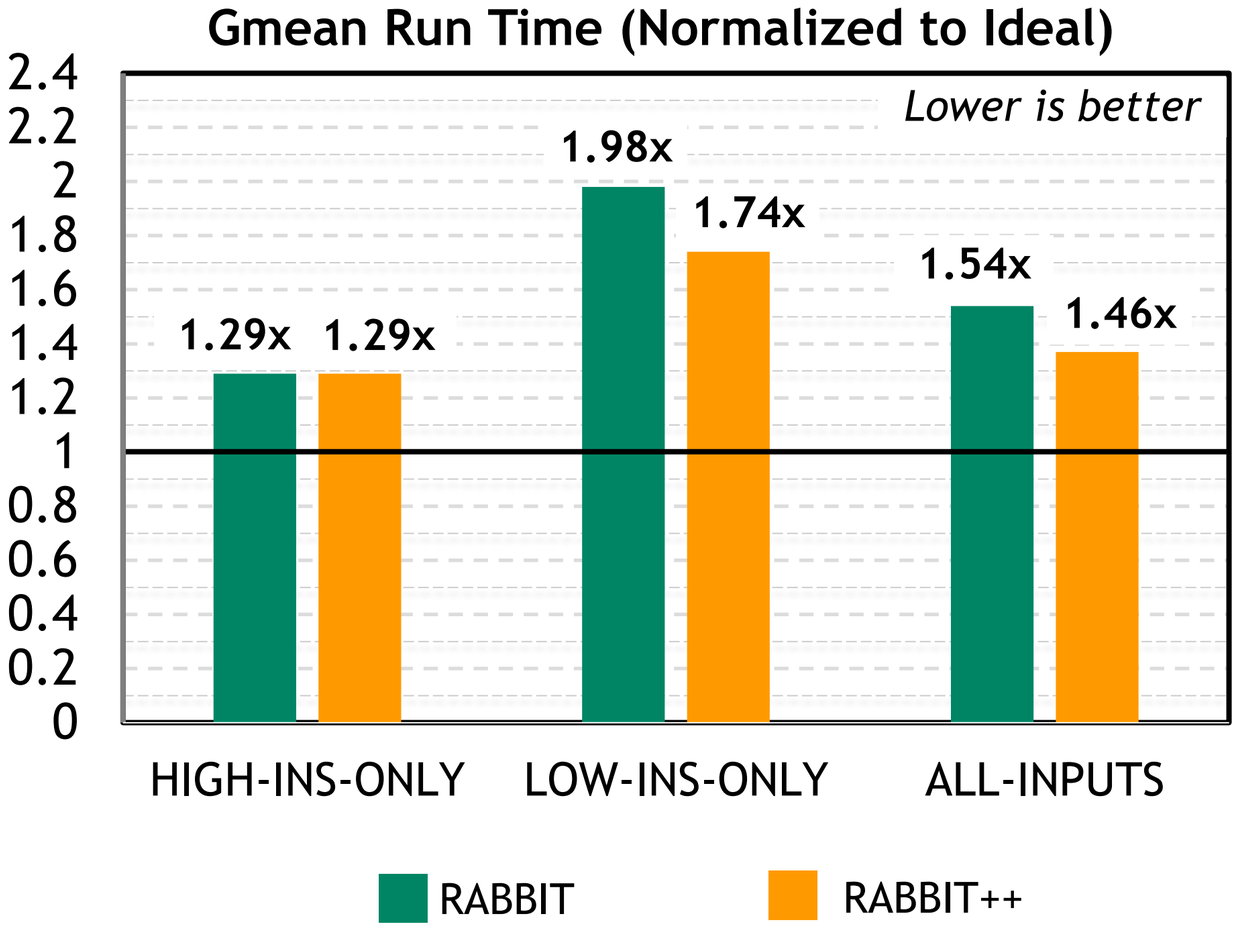


Ideal Traffic = Algorithmic Minimum DRAM Transfers

PERFORMANCE IMPROVEMENTS WITH RABBIT++

cuSPARSE SpMV on NVIDIA A6000 GPU



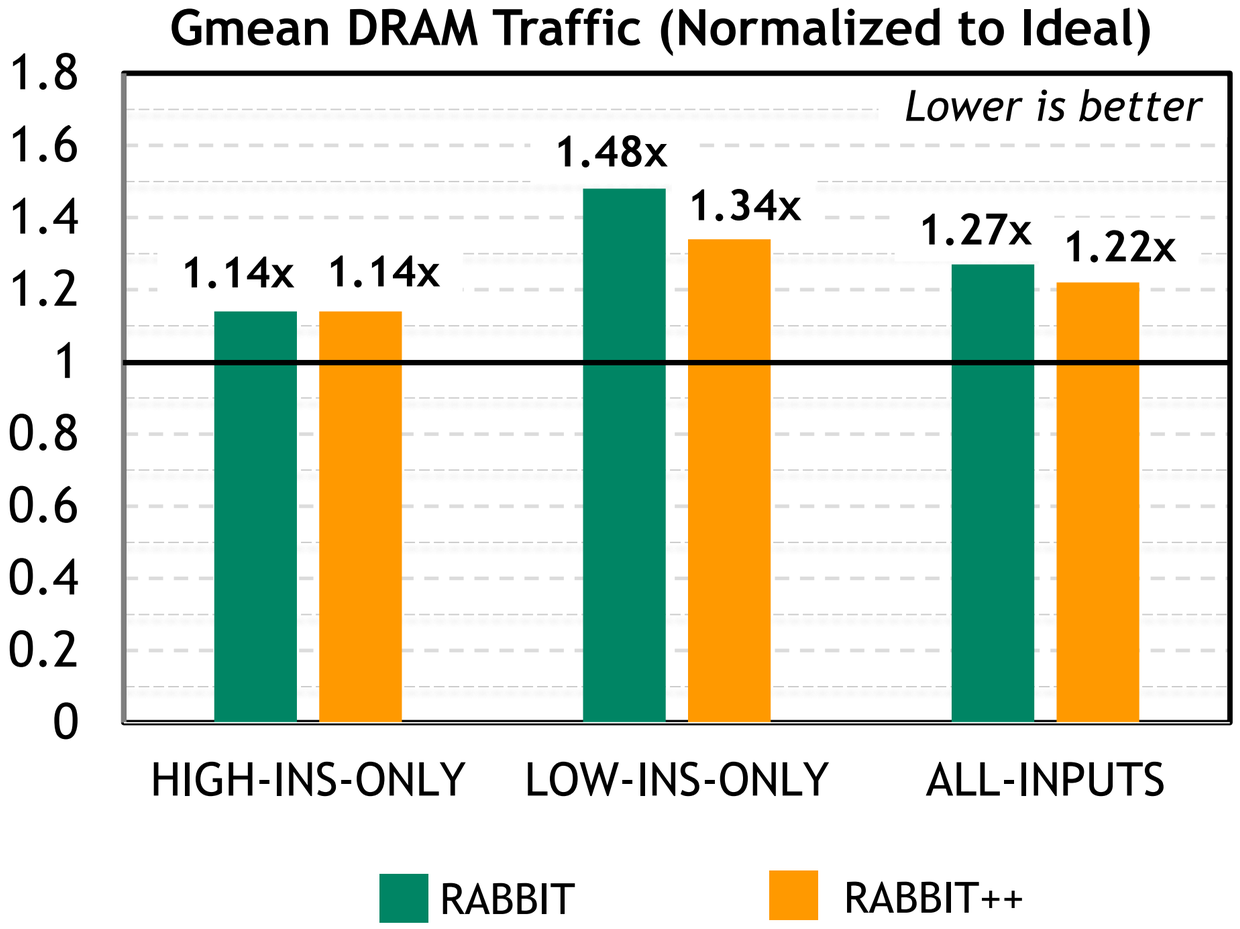


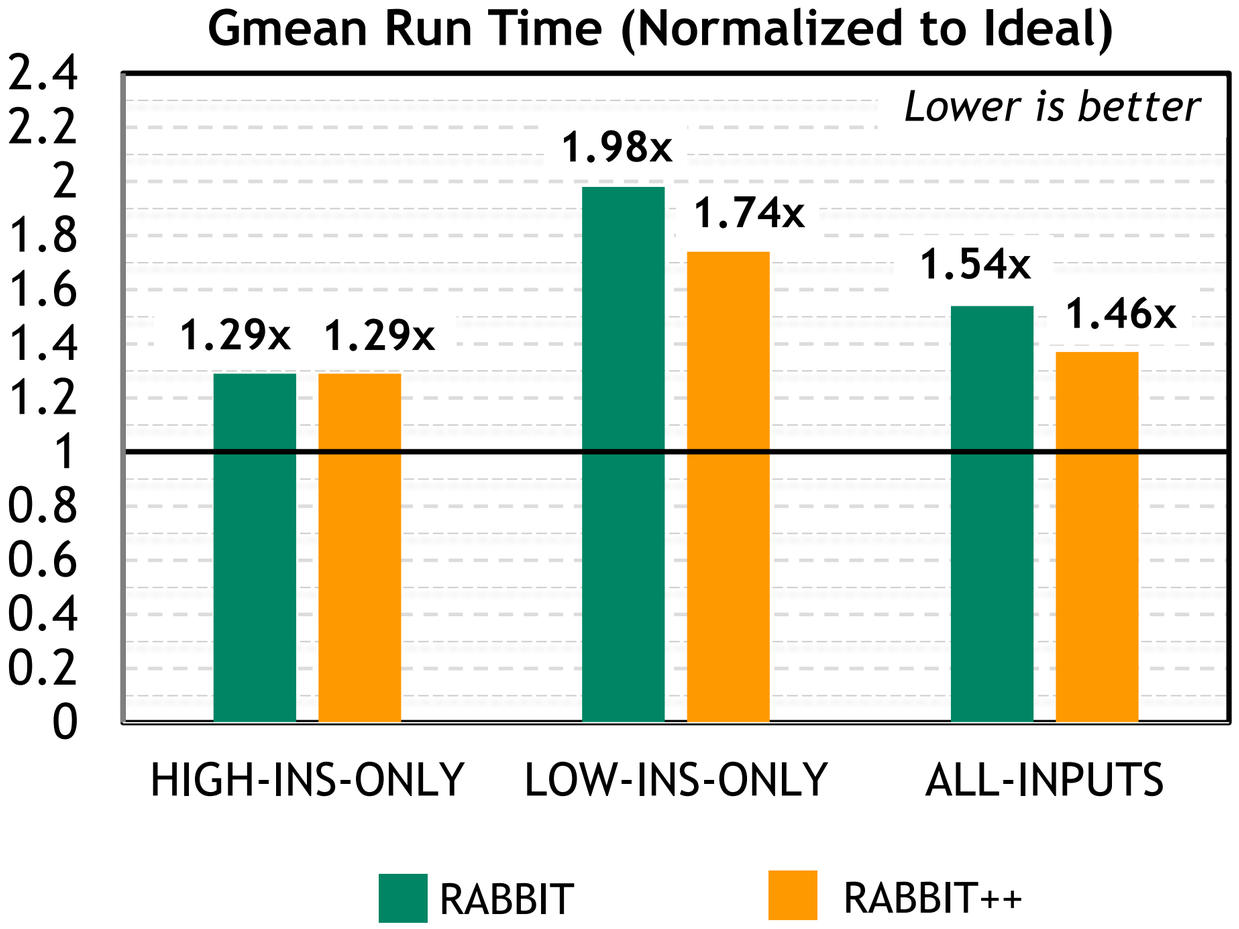
Ideal Traffic = Algorithmic Minimum DRAM Transfers

$$\text{Ideal Run Time} = \frac{\text{Algorithmic Min Traffic}}{\text{Peak DRAM Bandwidth}}$$

PERFORMANCE IMPROVEMENTS WITH RABBIT++

cuSPARSE SpMV on NVIDIA A6000 GPU





RABBIT++ improves SpMV performance over RABBIT by up to 1.6x

MORE DETAILS IN THE PAPER

- RABBIT++ offers DRAM traffic reductions close to Belady's Optimal cache replacement policy
- RABBIT++ brings multiple compressed representations and kernels closest to hardware limits
- Preprocessing overheads of RABBIT(++)

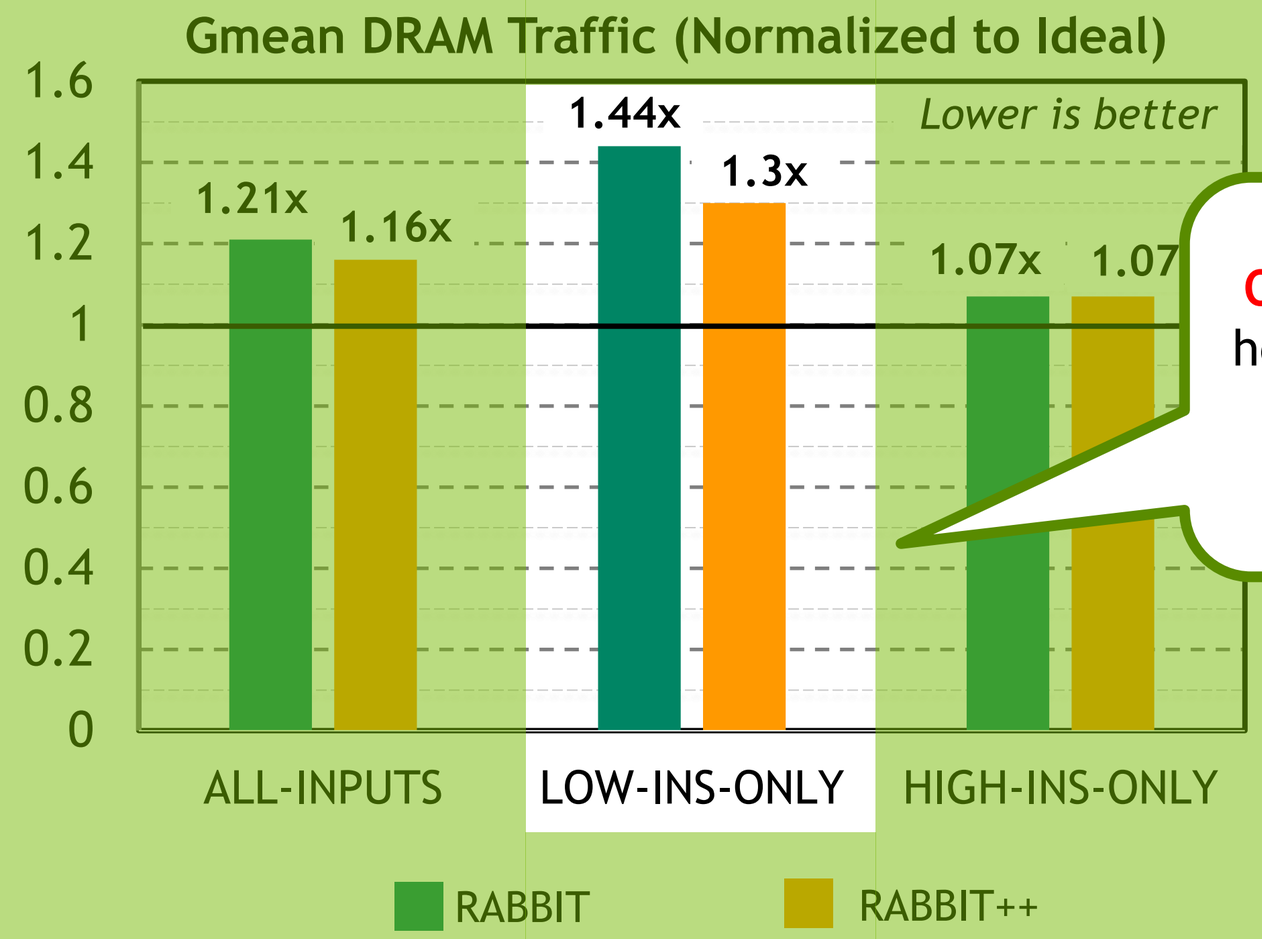
COMMUNITY-BASED MATRIX REORDERING FOR SPARSE LINEAR ALGEBRA OPTIMIZATION

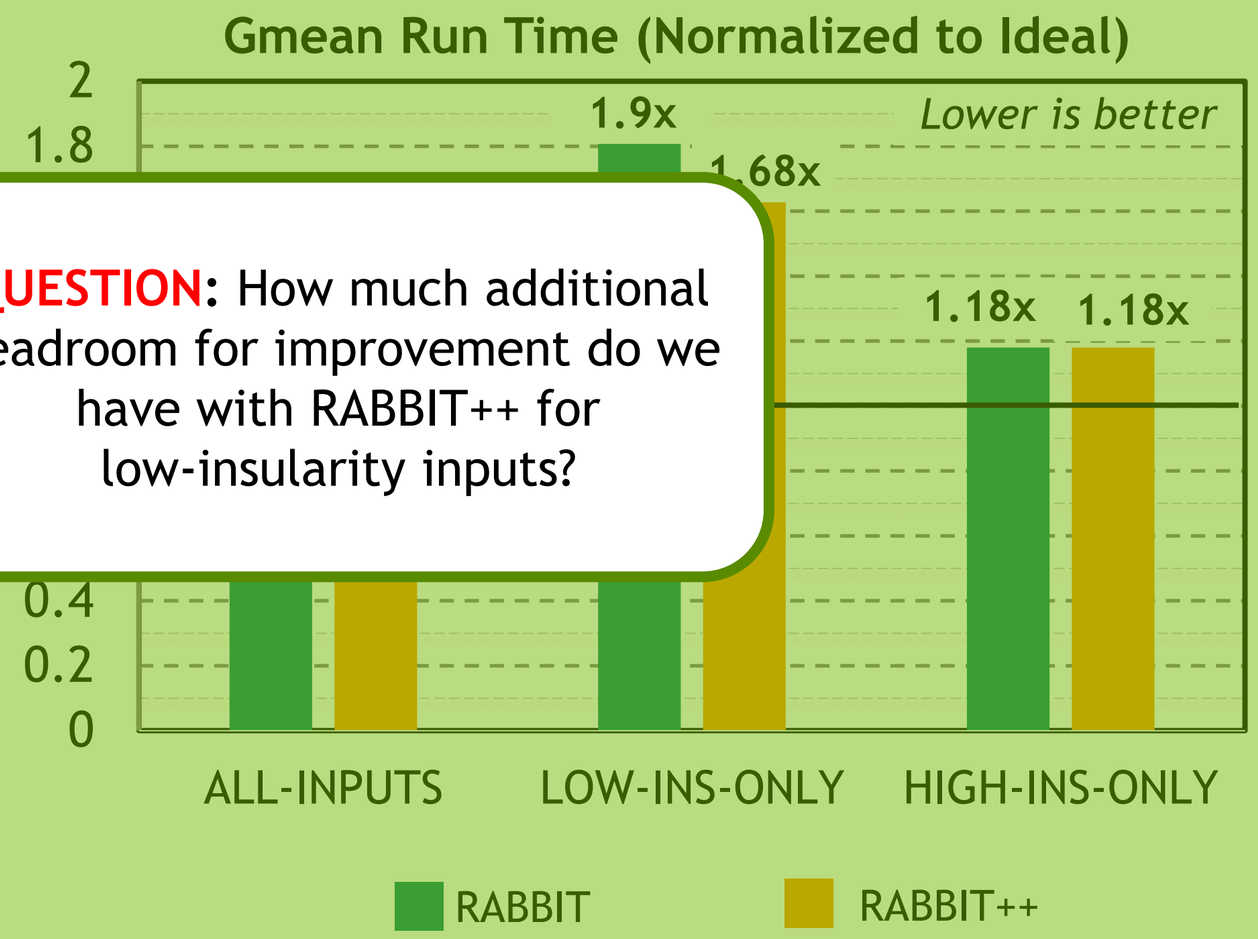
Vignesh Balaji Neal Crago Aamer Jaleel Stephen W. Keckler

BACKUP SLIDES

PERFORMANCE IMPROVEMENTS WITH RABBIT++

cuSPARSE SpMV on NVIDIA A6000 GPU



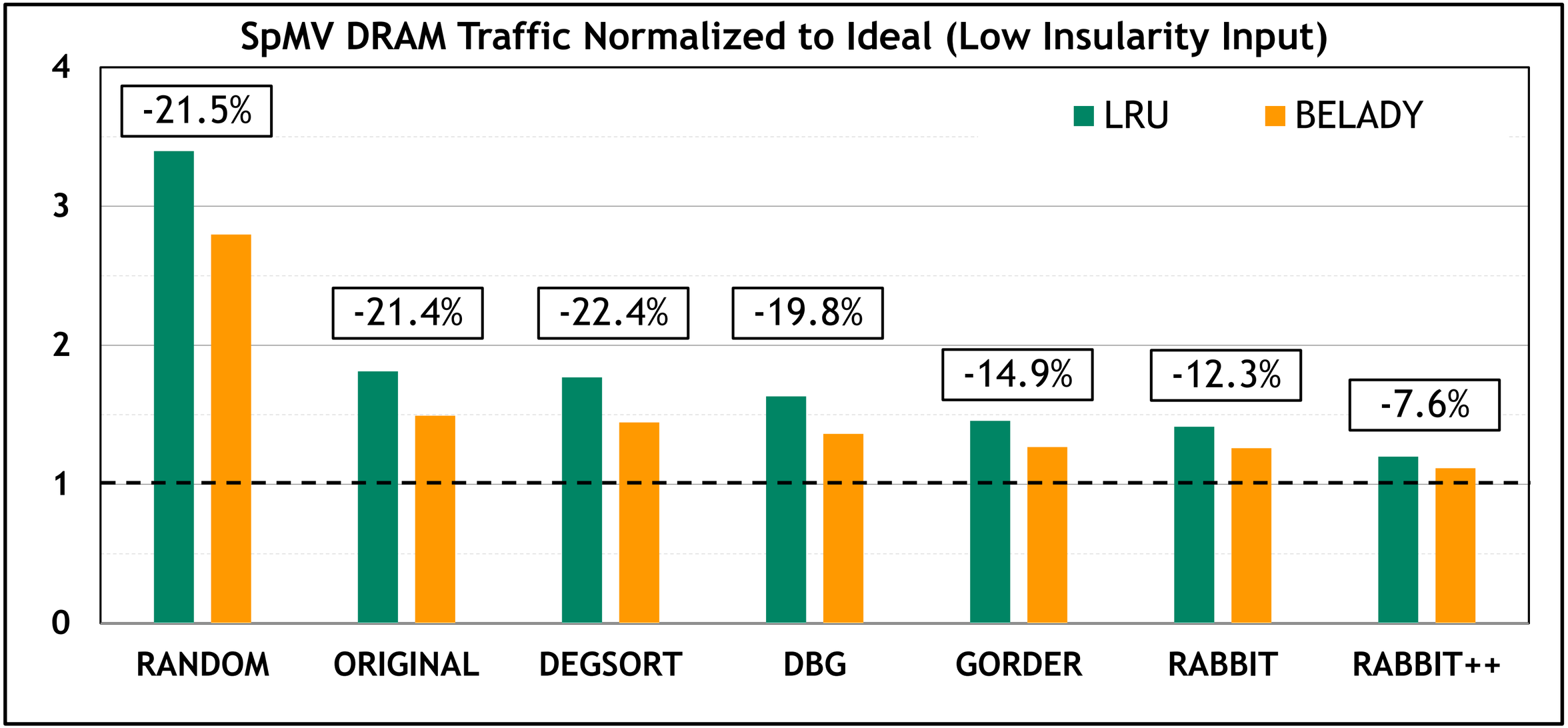


QUESTION: How much additional headroom for improvement do we have with RABBIT++ for low-insularity inputs?

RABBIT++ improves SpMV performance over RABBIT by up to 1.57x

HOW FAR IS RABBIT++ FROM OPTIMAL?

Experiments on a L2 cache simulator



RABBIT++ ACROSS DIFFERENT CUSPARSE KERNELS/FORMATS

TABLE IV: **Run time (normalized to ideal) across different cuSPARSE kernels:** *RABBIT++ consistently improves upon RABBIT's ability to bring kernels closer to peak performance.*

	SpMV-COO			SpMM-CSR-4			SpMM-CSR-256		
	ALL	*I <0.95*	*I >= 0.95*	*ALL*	*I <0.95*	*I >= 0.95*	*ALL*	*I <0.95*	*I >= 0.95*
RANDOM	5.37×	4.94×	5.97×	29.33×	32.17×	26.07×	139.3×	196.6×	75.13×
ORIGINAL	1.84×	2.1×	1.55×	5.97×	8.92×	3.58×	26.81×	43.79×	10.99×
RABBIT	1.49×	1.73×	**1.23×**	4.31×	7.39×	**2.18×**	20.32×	50.3×	**3.91×**
RABBIT++	**1.4×**	**1.55×**	**1.23×**	**3.79×**	**5.85×**	**2.18×**	**18.7×**	**43.97×**	3.95×

PREPROCESSING COSTS

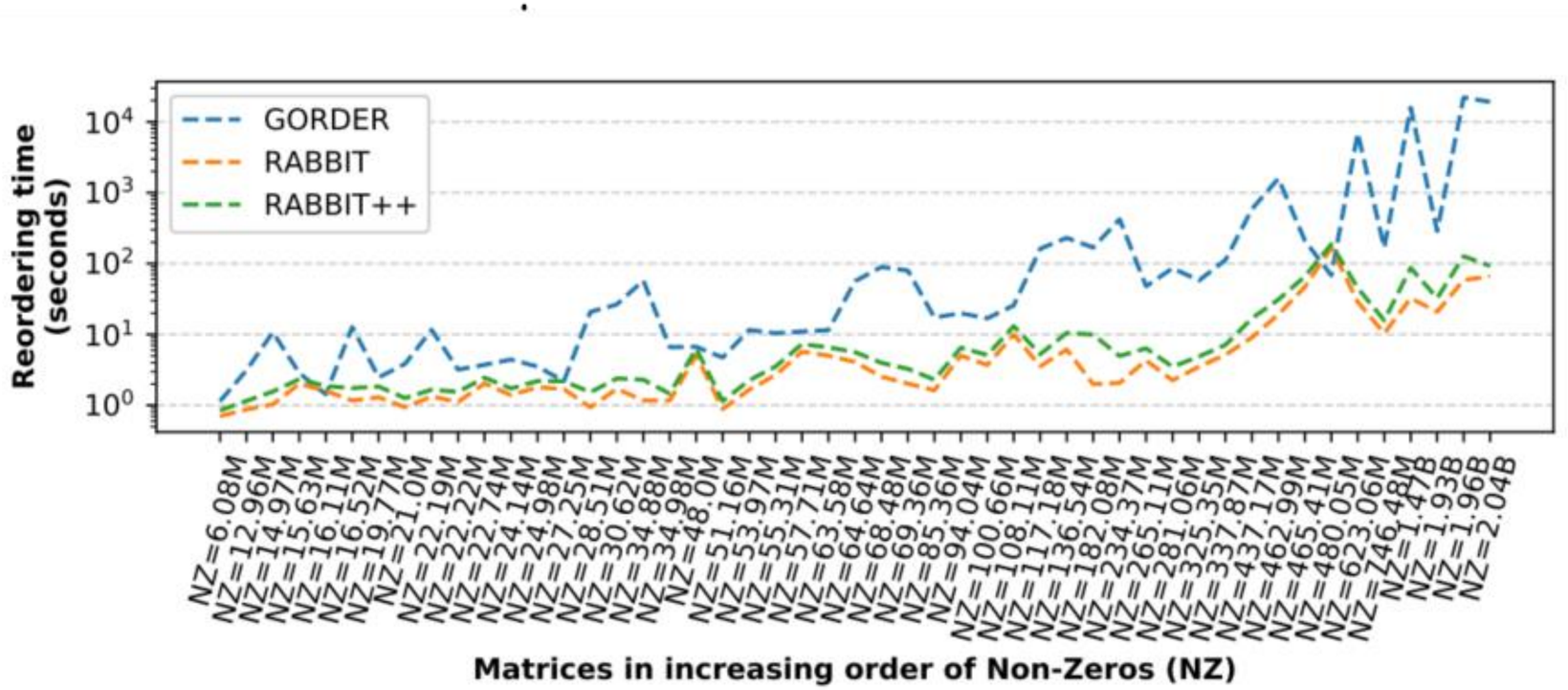


Fig. 9: **Matrix reordering time as the matrix size increases:** *Compared to GORDER, both RABBIT and RABBIT++ are more practical matrix reordering solutions.*

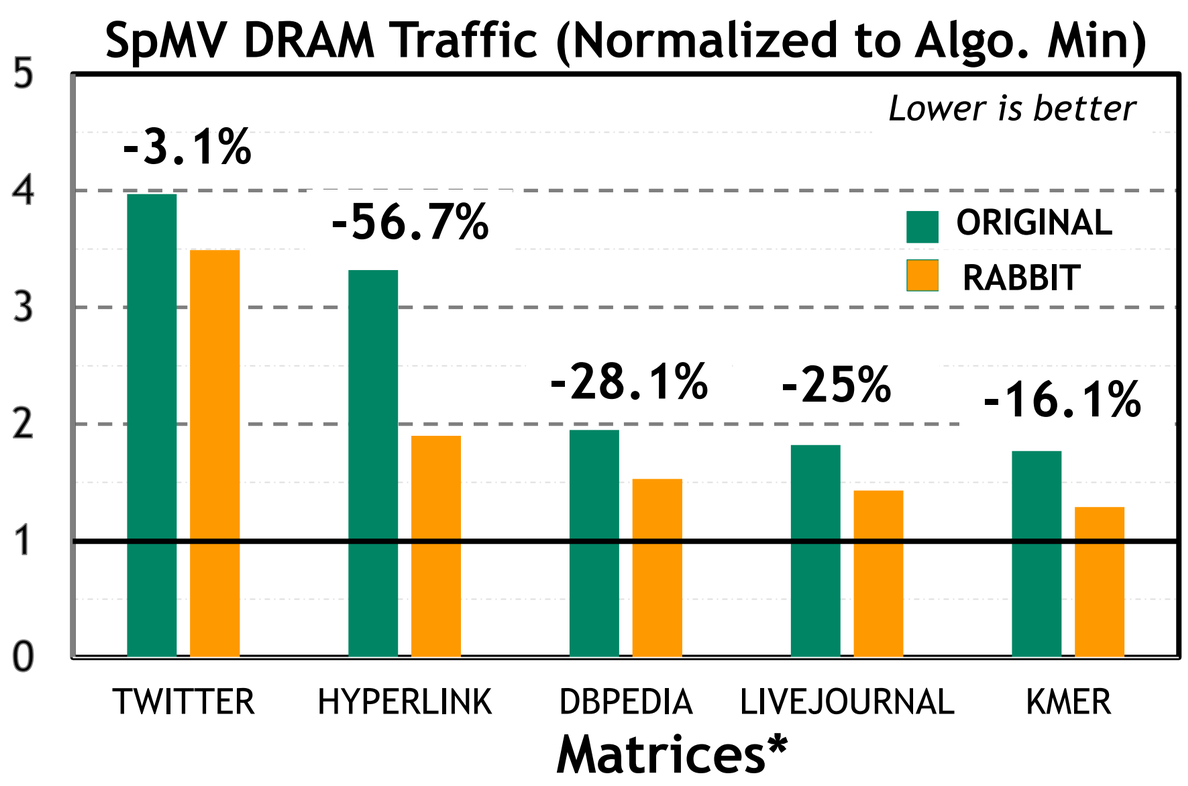
EVALUATION METHODOLOGY

HARDWARE: NVIDIA A6000 GPU

L2 Cache	DRAM Bandwidth	Mem Capacity
6MB	768GB/s	48GB

SOFTWARE: NVIDIA cuSPARSE library (v11.8)

INPUTS:



Locality improvement from reordering is sensitive to matrices

PROBLEM: Prior work on reordering were evaluated on a small number (<10) of arbitrarily-selected matrices

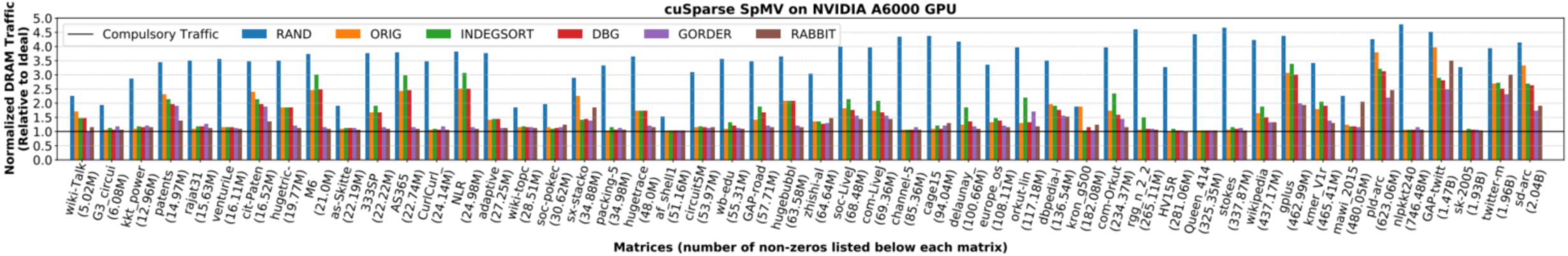
**Subset of matrices evaluated*

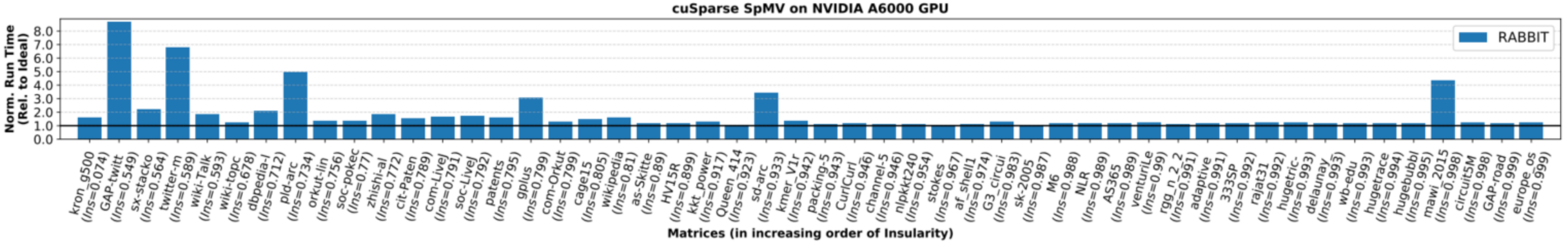
SPMV KERNEL

Algorithm 1 SpMV kernel with sparse matrix in CSR format

```
1: Y ← 0
2: parfor row in [0, |Rows|) do
3:     rowStart ← A.rowOffsets[row]
4:     rowEnd ← A.rowOffsets[row + 1]
5:     for i in [rowStart, rowEnd) do
6:         Y[row] += A.values[i] * X[A.coords[i]]
```

DRAM TRAFFIC AND PERF RESULTS ACROSS ALL INPUTS





DIFFERENT ALTERNATIVES FOR RABBIT++

TABLE II: **Design space of RABBIT modifications:** *SpMV run time (normalized to ideal) when applying different combinations of RABBIT modifications (Figure 5).*

	Without Insular Nodes Grouped			**With Insular Nodes Grouped**		
	ALL-MATS	*INS <0.95*	*INS >= 0.95*	*ALL-MATS*	*INS <0.95*	*INS >= 0.95*
RABBIT	1.54×	1.81×	1.25×	1.49×	1.70×	1.25×
RABBIT+HUBSORT	1.63×	1.89×	1.35×	1.57×	1.86×	1.26×
RABBIT+HUBGROUP	1.48×	1.65×	1.29×	**1.46**×	1.65×	1.25×

RABBIT++ TRAFFIC REDUCTIONS

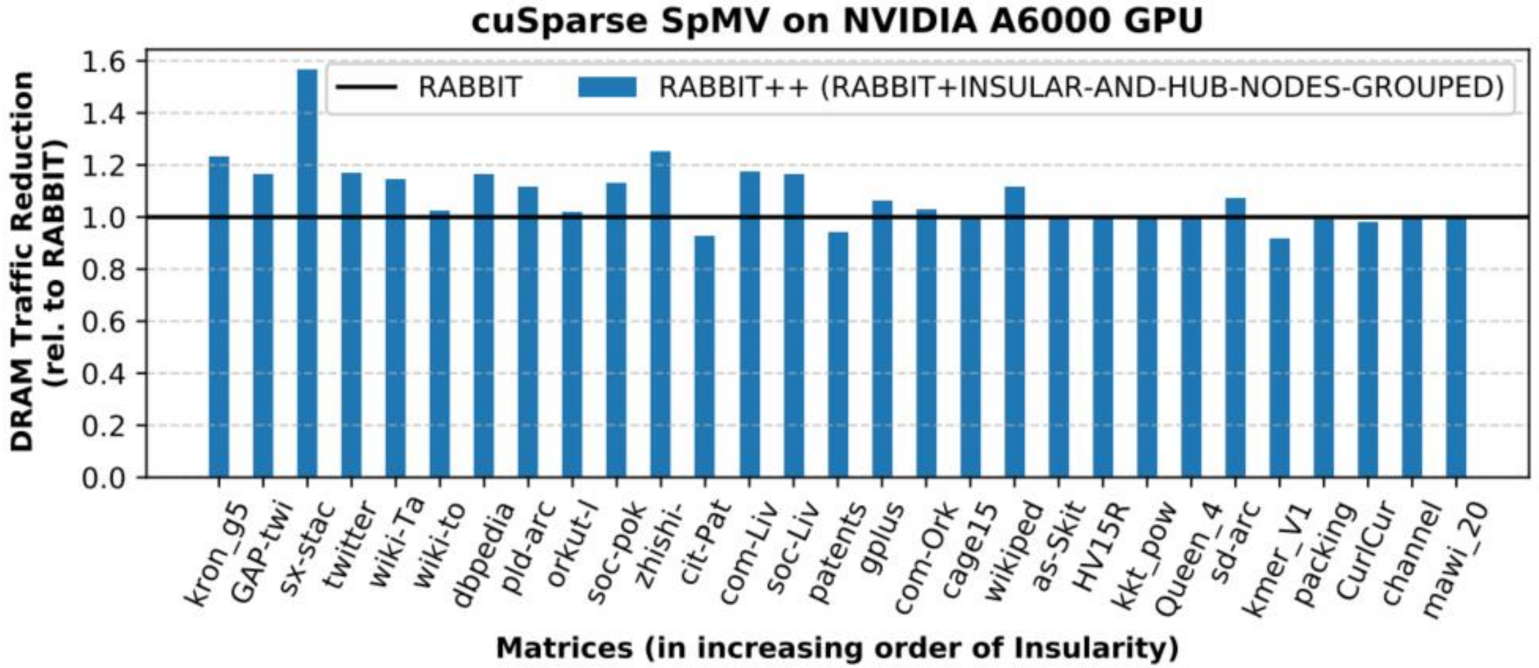


Fig. 7: **Reduction in SpMV's DRAM traffic with RABBIT++:** *In the interest of space, we only include results for matrices with Insularity* < 0.95*). For matrices with Insularity* ≥ 0.95*, RABBIT++'s DRAM traffic is within 1% of RABBIT.*

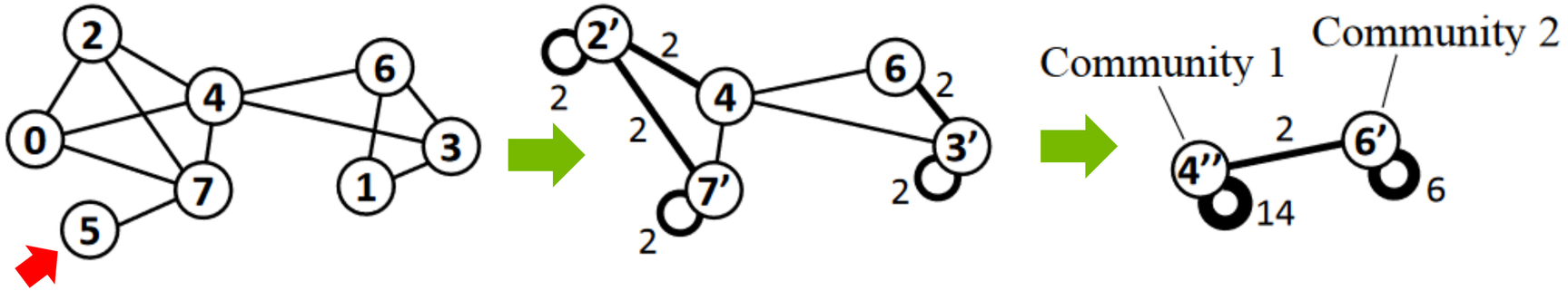
RABBIT++ ACROSS DIFFERENT CUSPARSE KERNELS/FORMATS

TABLE IV: **Run time (normalized to ideal) across different cuSPARSE kernels:** ***RABBIT++ consistently improves upon RABBIT's ability to bring kernels closer to peak performance.***

	SpMV-COO			**SpMM-CSR-4**			**SpMM-CSR-256**		
	ALL	*I <0.95*	*I >= 0.95*	*ALL*	*I <0.95*	*I >= 0.95*	*ALL*	*I <0.95*	*I >= 0.95*
RANDOM	5.37×	4.94×	5.97×	29.33×	32.17×	26.07×	139.3×	196.6×	75.13×
ORIGINAL	1.84×	2.1×	1.55×	5.97×	8.92×	3.58×	26.81×	43.79×	10.99×
RABBIT	1.49×	1.73×	**1.23×**	4.31×	7.39×	**2.18×**	20.32×	50.3×	**3.91×**
RABBIT++	**1.4×**	**1.55×**	**1.23×**	**3.79×**	**5.85×**	**2.18×**	**18.7×**	**43.97×**	3.95×

COMMUNITY-BASED GRAPH REORDERING

High-level Overview of RABBIT

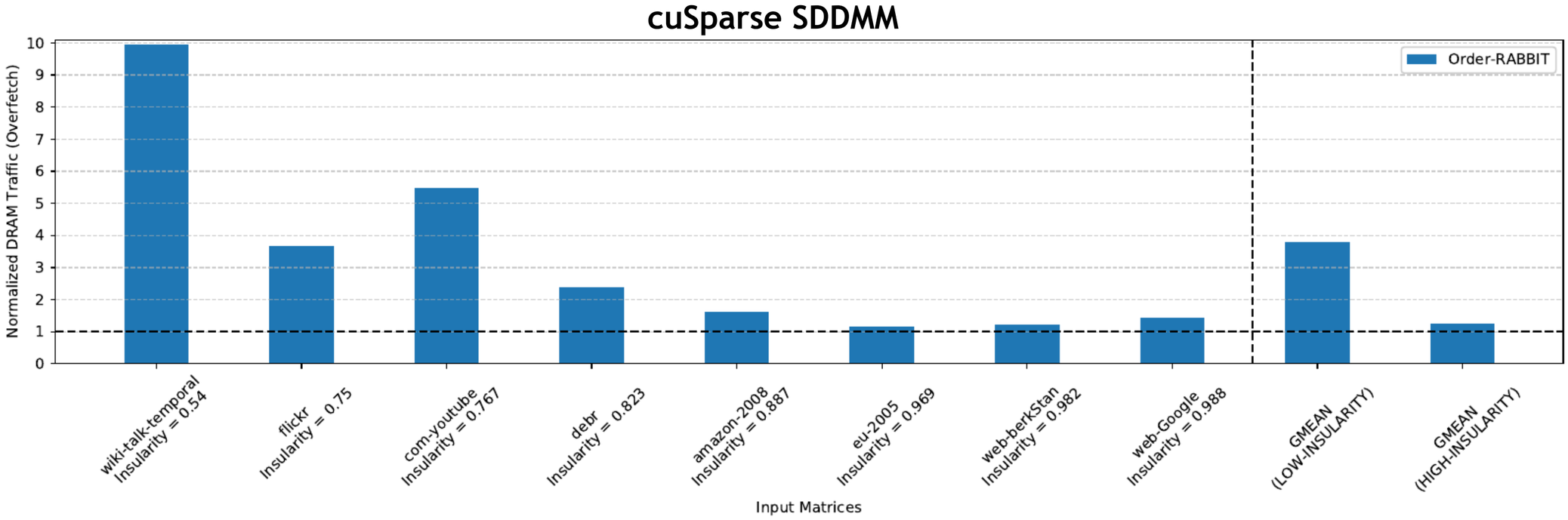


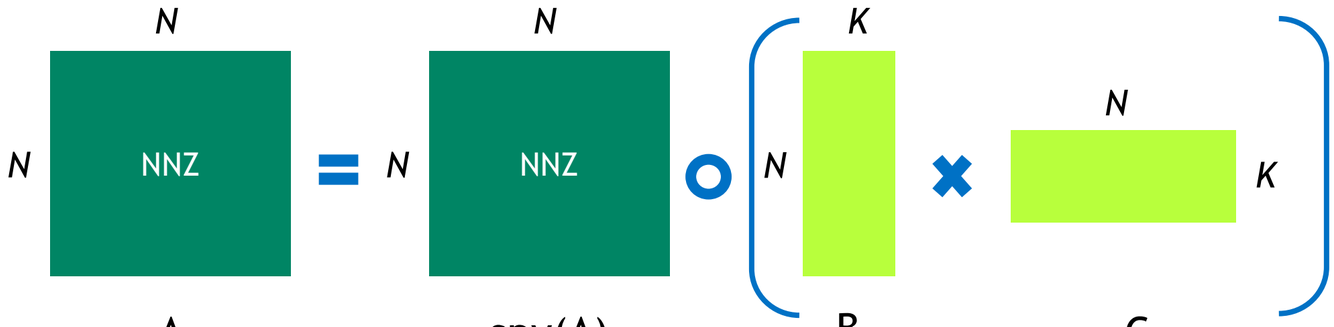
Incremental aggregation on the graph

$$\Delta Q(u,v) = 2\left\{\frac{w_{uv}}{2m} - \frac{d(u)d(v)}{(2m)^2}\right\}$$

Figure from "Rabbit Order: Just-in-time Parallel Reordering for Fast Graph Analysis" IPDPS 2016

RABBIT IMPROVES LOCALITY ACROSS MULTIPLE KERNELS





Insularity

A
(Sparse Matrix)

spy(A)
(Sparse Matrix)

B
(Dense Matrix)

C
(Dense Matrix)

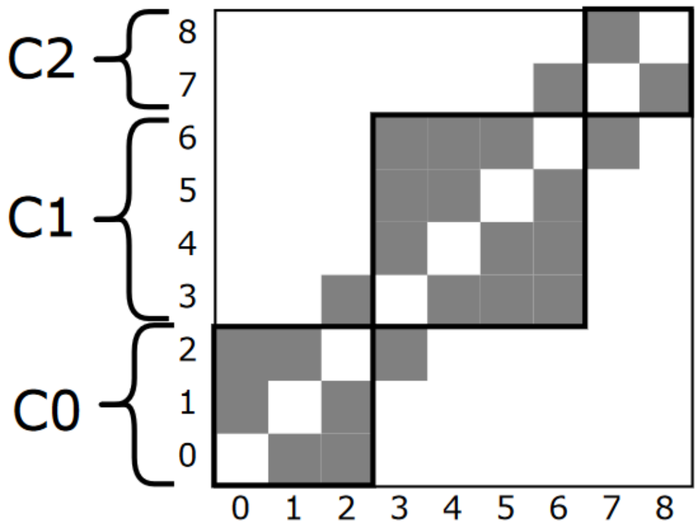
COMPARING EXISTING REORDERING TECHNIQUES

Reordering Technique*	Structural Property Targeted	Intuition
Degree Sorting (DEGSORT)	Power-law degree distribution	Assign IDs in decreasing order of vertex degrees
Degree Based Grouping (DBG)	Power-law degree distribution	Assign contiguous IDs to vertices in the same degrees bucket
RABBIT	Community Structure	Assign contiguous IDs to vertices in the same community
GORDER	Power-Law + Community	Assign contiguous IDs to vertices with a strong overlap in neighborhoods

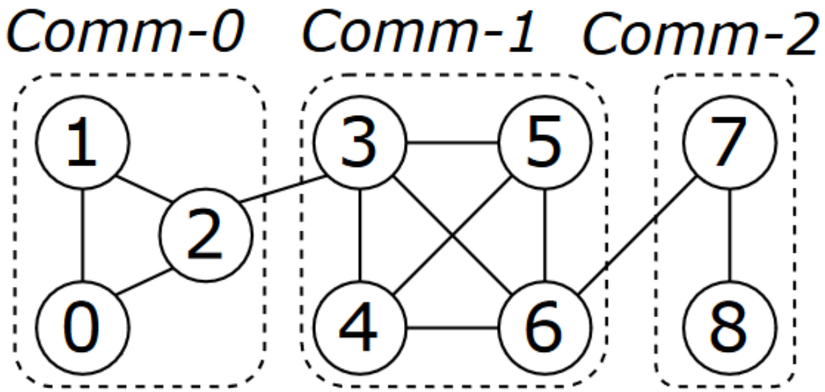
**Techniques organized in increasing order of reordering complexity*

A METRIC TO QUANTIFY BENEFITS WITH RABBIT

Insularity



From a locality perspective, we need **highly self-contained and small** communities

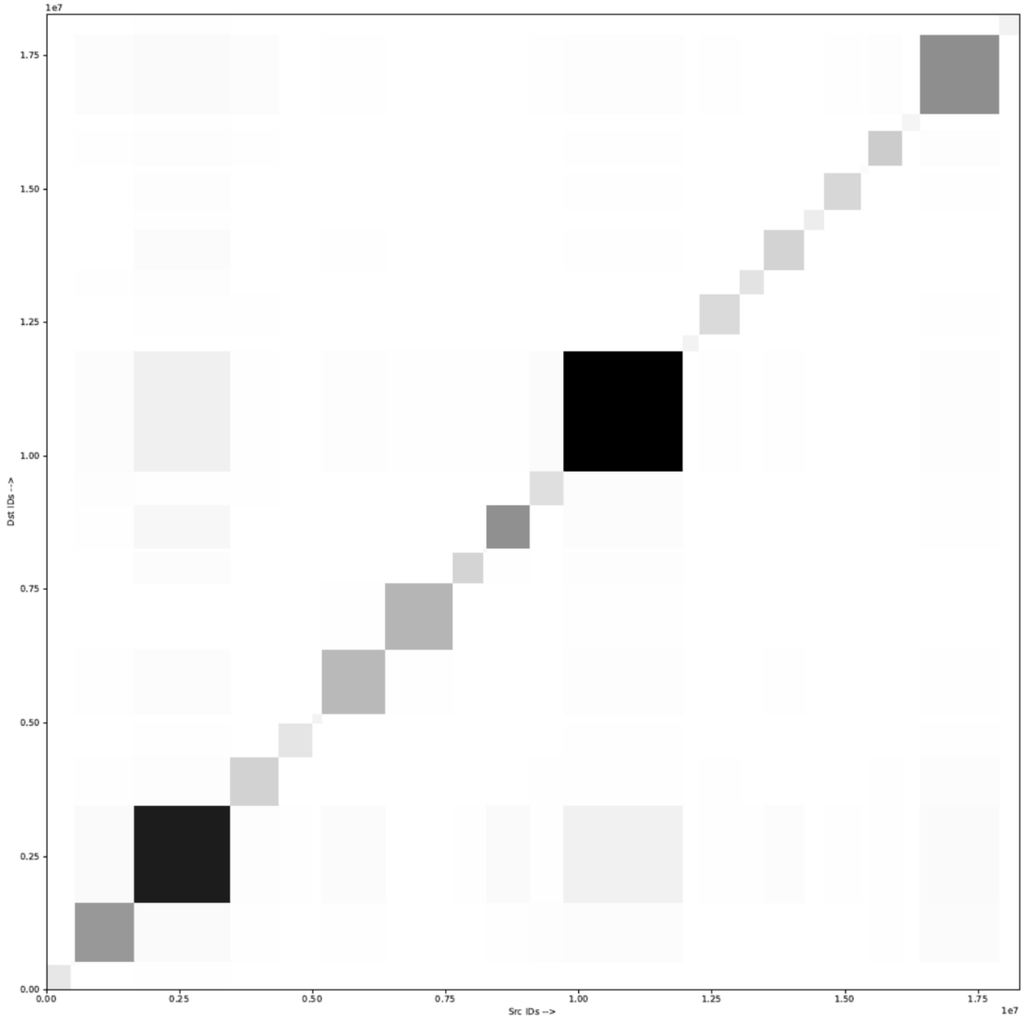


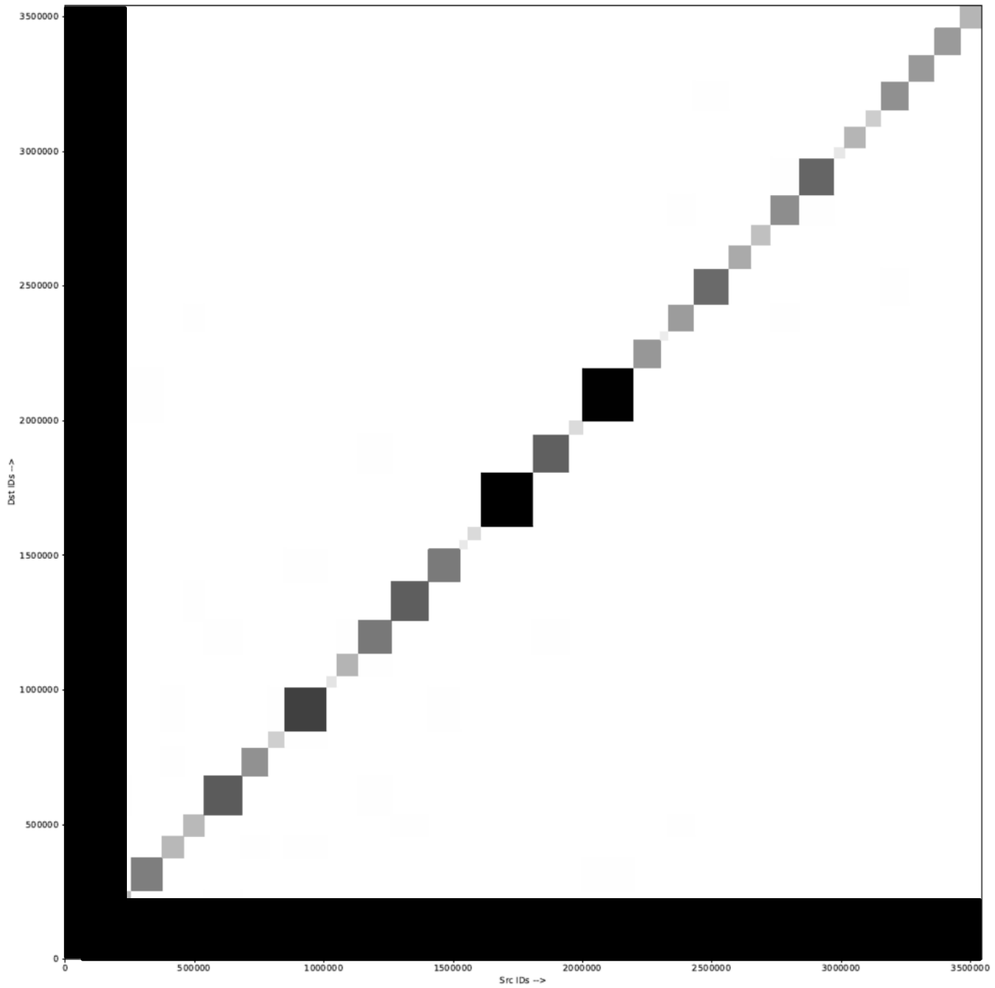
$$\text{Insularity} = \frac{Intra{-}Comm{-}Edges}{Total{-}Comm{-}Edges} = \frac{20}{24} = 0.83$$

SpMV Run Time (normalized to ideal) has a strong inverse correlation with insularity (ρ = -0.74)

ENHANCING RABBIT

**Cartoon representation*





Insular Nodes

Hubs

RABBIT

RABBIT++

MATRIX REORDERING ACROSS DIFFERENT GPUS