SMASH

Co-Designing Software Compression and Hardware-Accelerated Indexing for Efficient Sparse Matrix Operations

Konstantinos Kanellopoulos, Nandita Vijaykumar, Christina Giannoula,
Roknoddin Azizi, Skanda Koppula, Nika Mansouri Ghiasi,
Taha Shahroodi, Juan Gomez Luna, Onur Mutlu

SAFARI

ETH zürich

Carnegie Mellon University

Executive Summary

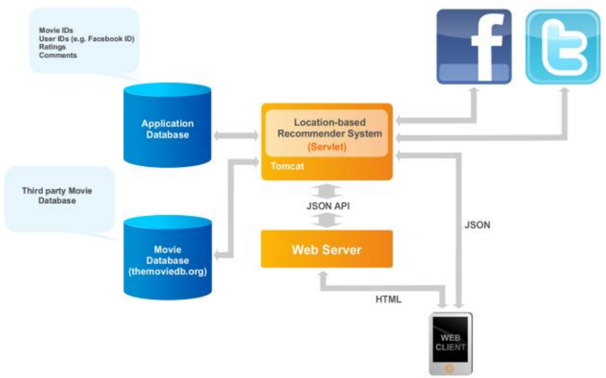
- Many important workloads heavily make use of **sparse matrices**
  - Matrices are **extremely sparse**
  - **Compression** is essential to avoid storage/computational overheads
- Shortcomings of existing compression formats
  - **Expensive discovery of the positions of non-zero elements or**
  - **Narrow applicability**
- **SMASH**: hardware/software cooperative mechanism for efficient sparse matrix storage and computation
  - ***Software***: Efficient compression scheme using **a hierarchy of bitmaps**
  - ***Hardware***: Hardware unit interprets the **bitmap hierarchy** and accelerates indexing
- Performance improvement: **38% and 44%** for **SpMV** and **SpMM** over the widely used CSR format
- SMASH is **highly efficient, low-cost** and **widely applicable**

Presentation Outline

1. Sparse Matrix Basics
2. Compression Formats & Shortcomings
3. SMASH
4. Evaluation
5. Conclusion

Sparse Matrix Operations are Widespread Today

Recommender Systems



- Collaborative Filtering

Graph Analytics

- PageRank
- Breadth First Search
- Betweenness Centrality

Neural Networks

- Sparse DNNs
- Graph Neural Networks

Real World Matrices Have High Sparsity

0.0003%
non-zero elements

2.31%
non-zero elements

Sparse matrix compression is essential to enable efficient storage and computation

Presentation Outline

Limitations of Existing Compression Formats

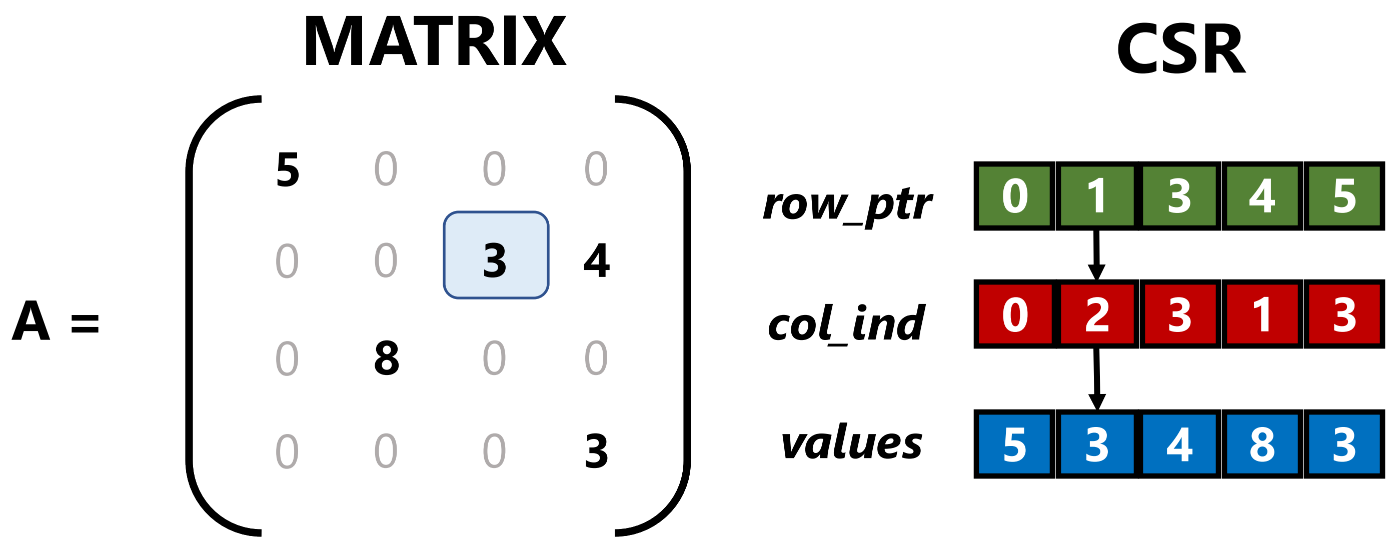
❶

General formats optimize for storage → Expensive discovery of the positions of non-zero elements

Widely Used Format: Compressed Sparse Row

- Used in multiple **libraries** & **frameworks**:
  Intel MKL, TACO, Ligra, Polymer
- Stores **only the non-zero elements**
  of the sparse matrix
- Provides **high compression ratio**

Basics of CSR: Indexing Overhead



Multiple instructions are needed to discover the position of a non-zero element

Requires multiple data-dependent memory accesses

Indexing Overhead in Sparse Kernels

Sparse Matrix Vector Multiplication (SpMV)

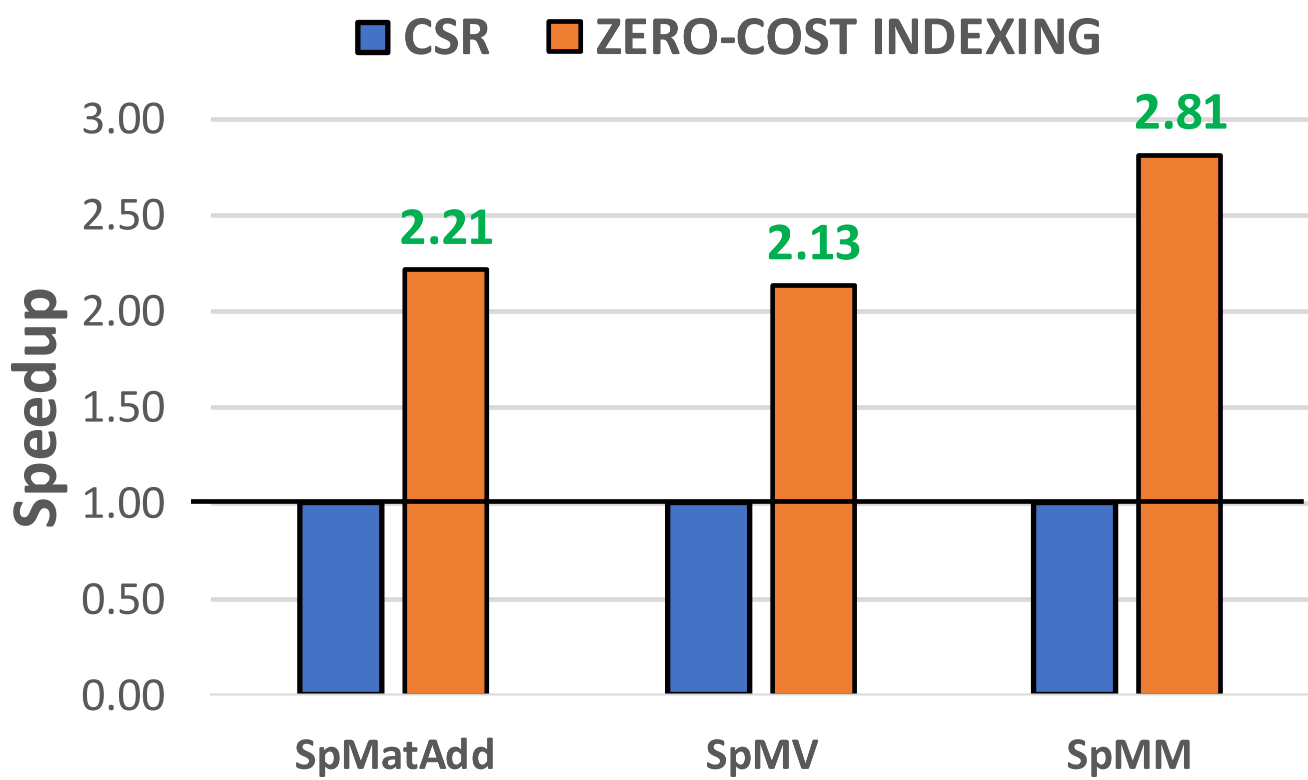
Indexing for every non-zero element of the sparse matrix to multiply with the corresponding element of the vector.

Sparse Matrix Matrix Multiplication (SpMM)

Index matching for every inner product between the 2 sparse matrices.

Indexing is expensive for major sparse matrix kernels

Performing Indexing with Zero Cost



Reducing the cost of indexing can accelerate sparse matrix operations

Limitations of Existing Compression Formats

❶

General formats optimize for storage → **Expensive** discovery of the positions of non-zero elements

❷

Specialized formats assume <u>specific matrix structures</u> and patterns (e.g., diagonals) → **Narrow applicability**

Our Goal

Design a sparse matrix compression mechanism that:

- **Minimizes** the indexing **overheads**
- Can be used across a **wide range** of sparse matrices and sparse matrix operations
- Enables **high compression ratio**

Presentation Outline

1. Sparse Matrix Basics
2. Compression Formats & Shortcomings
3. SMASH
    - Software Compression Scheme
    - Hardware Acceleration Unit
    - Cross-layer Interface
4. Evaluation
5. Conclusion

SMASH: Key Idea

Hardware/Software cooperative mechanism:

- Enables **highly-efficient** sparse matrix compression and computation
- **General** across a diverse set of sparse matrices and sparse matrix operations

Software

Efficient compression using a Hierarchy of Bitmaps

Hardware

Unit that scans bitmaps to accelerate indexing

SMASH ISA

Presentation Outline

1. Sparse Matrix Basics
2. Compression Formats & Shortcomings
3. SMASH
    - Software Compression Scheme
    - Hardware Acceleration Unit
    - Cross-layer Interface
4. Evaluation
5. Conclusion

SMASH: Software Compression Scheme

Software

Efficient compression using a Hierarchy of Bitmaps

- Encodes the positions of non-zero elements using bits to maintain **low storage overhead**

SMASH: Software Compression Scheme

BITMAP:

Encodes if a block of the matrix contains any non-zero element

MATRIX

NZ	ZEROS	ZEROS	ZEROS
ZEROS	NZ	ZEROS	ZEROS
ZEROS	ZEROS	ZEROS	ZEROS
ZEROS	ZEROS	NZ	NZ

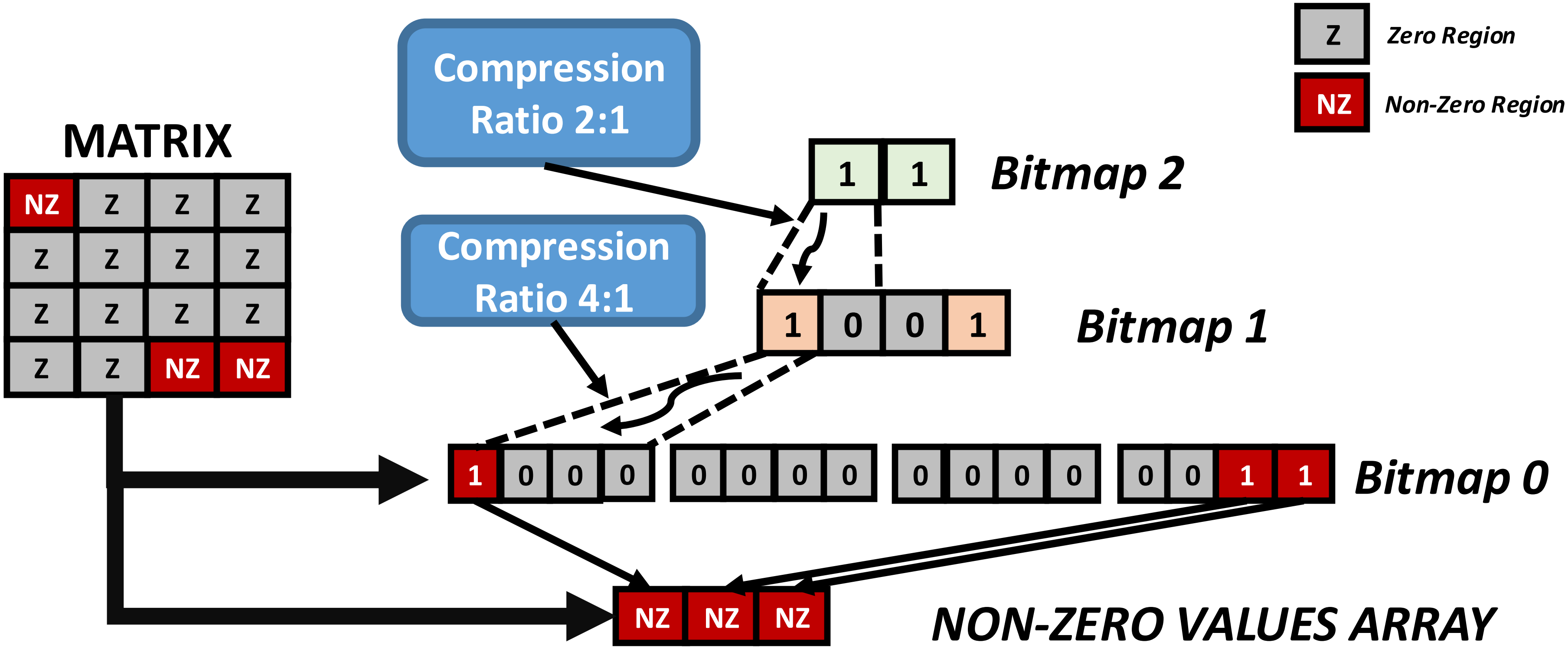
BITMAP

1	0	0	0
0	1	0	0
0	0	0	0
0	0	1	1

Might contain high number of zero bits

Idea: Apply the same encoding recursively to compress more effectively

Hierarchy of Bitmaps



Storage Efficient	Fast Indexing	Hardware Friendly

Presentation Outline

1. Sparse Matrix Basics
2. Compression Formats & Shortcomings
3. SMASH
    - Software Compression Scheme
    - Hardware Acceleration Unit
    - Cross-layer Interface
4. Evaluation
5. Conclusion

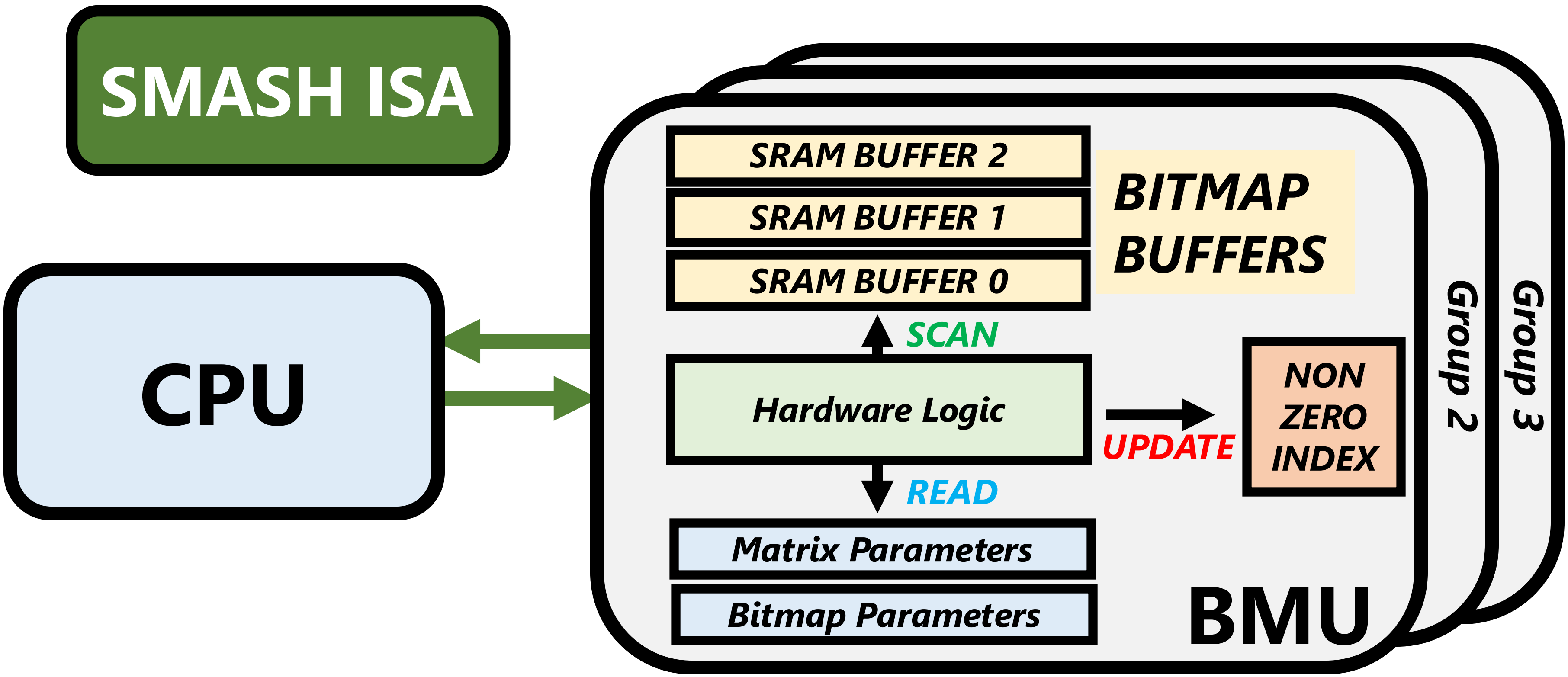
SMASH: Hardware Acceleration Unit

Hardware

Unit that scans bitmaps to accelerate indexing

- **Interprets** the **Bitmap Hierarchy** used in software
- **Buffers** the **Bitmap Hierarchy**
  and **scans** it using bitwise operations

Bitmap Management Unit (BMU)



Presentation Outline

SMASH: Cross-Layer Interface

Need for a cross-layer interface that enables software to <u>control the BMU</u>

- **Communicate** the **parameters** needed to calculate the index
- **Query** the BMU to **retrieve the index** of the next non-zero element

SMASH ISA

```
MATINFO
BMAPINFO
RDBMAP
PBMAP
RDIND
```

Enables SMASH to flexibly accelerate a diverse range of operations on any sparse matrix

Presentation Outline

Methodology

Simulator: *ZSim* Simulator [1]

Workloads:

- **Sparse Matrix Kernels**
  SpMV & SpMM from TACO [2]

- **Graph Applications**
  PageRank & Betweenness Centrality from Ligra [3]

Input datasets:

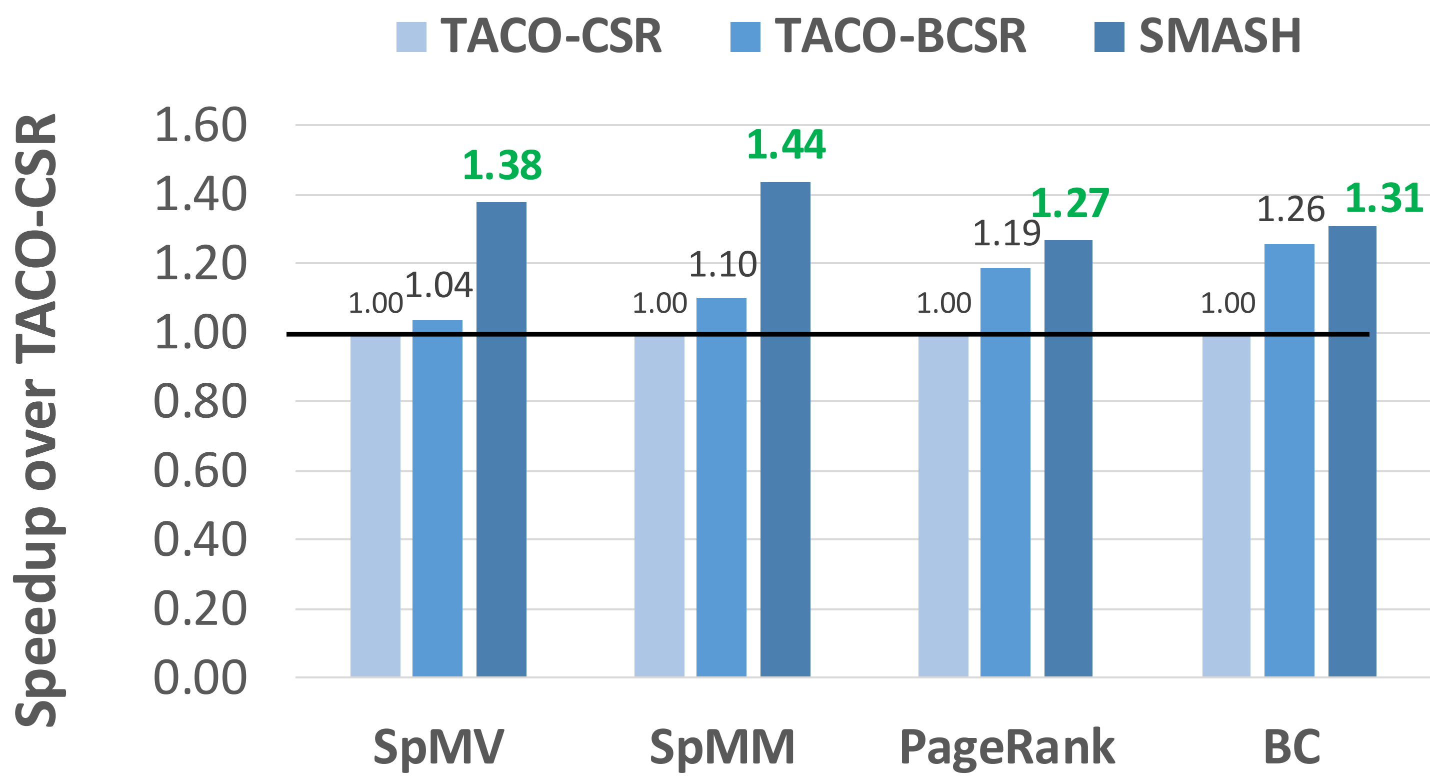
15 diverse sparse matrices & 4 graphs
from the Sparse Suite Collection [4]

[1] *Sanchez et al.* "ZSim: Fast and Accurate Microarchitectural Simulation of Thousand-Core Systems" ISCA'13
[2] *Kjolstad et al.* " taco: A Tool to Generate Tensor Algebra Kernels" ASE'17
[3] *Shun et al. "Ligra: A Lightweight Graph Processing Framework for Shared Memory" PPoPP'13*
[4] *Davis et al. "The University of Florida Sparse Matrix Collection" TOMS'11*

Evaluated Sparse Matrices

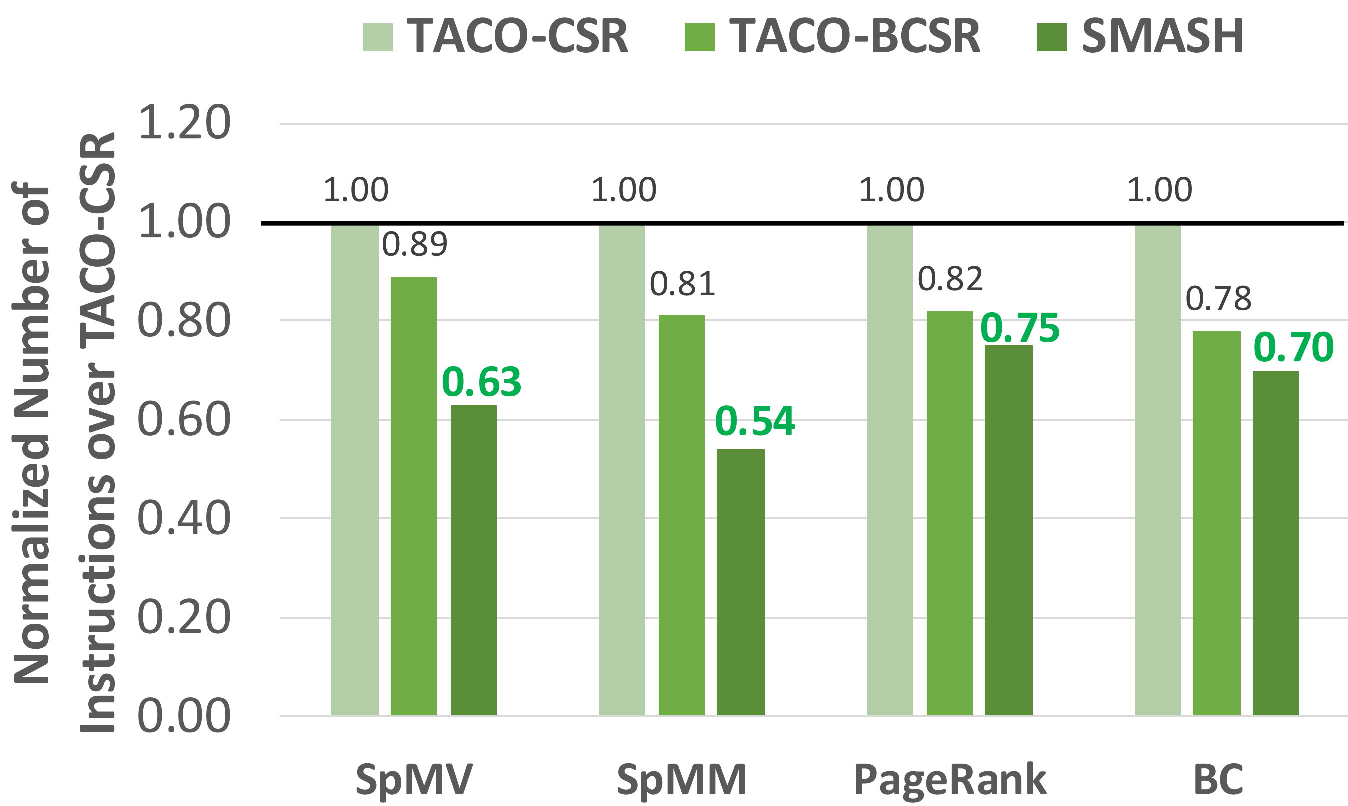
	Name	# Rows	Non-Zero Elements	Sparsity (%)
M1:	descriptor_xingo6u	20,738	73,916	0.01
M2:	g7jac060sc	17,730	183,325	0.06
M3:	Trefethen_20000	20,000	554,466	0.14
M4:	IG5-16	18,846	588,326	0.17
M5:	TSOPF_RS_b162_c3	15,374	610,299	0.26
M6:	ns3Da	20,414	1,679,599	0.40
M7:	tsyl201	20,685	2,454,957	0.57
M8:	pkustk07	16,860	2,418,804	0.85
M9:	ramage02	16,830	2,866,352	1.01
M10:	pattern1	19,242	9,323,432	2.52
M11:	gupta3	16,783	9,323,427	3.31
M12:	nd3k	9,000	3,279,690	4.05
M13:	human_gene1	22,283	24,669,643	4.97
M14:	exdata_1	6,001	2,269,500	6.30
M15:	human_gene2	14,340	18,068,388	8.79

Performance Improvement Using SMASH



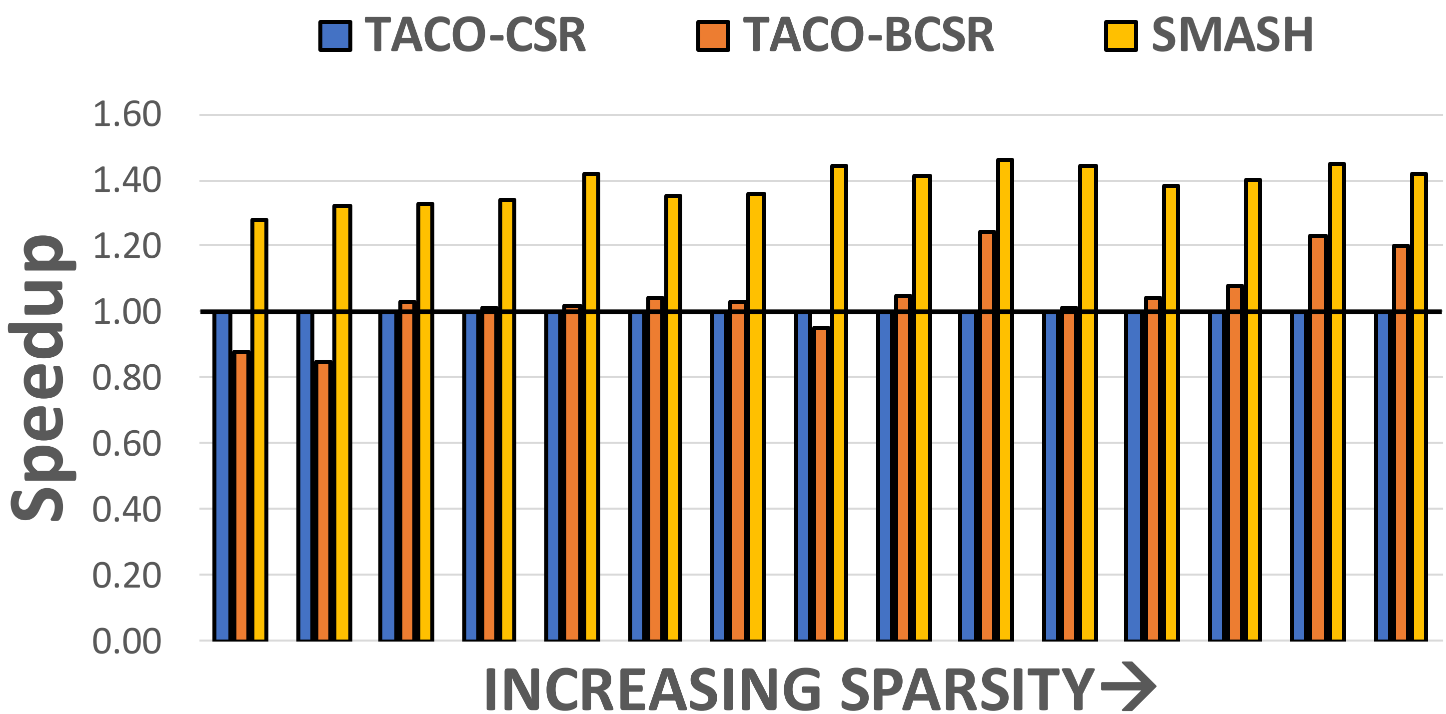
SMASH provides significant performance improvements over state-of-the-art formats

Number of Executed Instructions



SMASH significantly reduces the number of executed instructions

Sparsity Sweep for SpMV



SMASH provides speedups regardless of the sparsity of the matrix

Hardware Area Overhead

SMASH configuration

- Support for 4 matrices in the BMU
- 256 bytes / SRAM buffer
- 140 bytes for registers & counters

0.076% area overhead over an Intel Xeon CPU

SMASH incurs negligible area overhead

Other Results in the Paper

- Compression ratio sensitivity analysis
- Distribution of non-zero elements
- Detailed results for SpMM
- Conversion from CSR to SMASH overhead
- Software-only approaches executed in a real machine

Presentation Outline

1. Sparse Matrix Basics
2. Compression Formats & Shortcomings
3. SMASH
   - Software Compression Scheme
   - Hardware Acceleration Unit
   - Cross-layer Interface
4. Evaluation
5. Conclusion

Summary & Conclusion

- Many important workloads heavily make use of **sparse matrices**
  - Matrices are **extremely sparse**
  - **Compression** is essential to avoid storage/computational overheads
- Shortcomings of existing compression formats
  - **Expensive discovery of the positions of non-zero elements or**
  - **Narrow applicability**
- **SMASH**: hardware/software cooperative mechanism for efficient sparse matrix storage and computation
  - ***Software***: Efficient compression scheme using **a hierarchy of bitmaps**
  - ***Hardware***: Hardware unit interprets the **bitmap hierarchy** and accelerates indexing
- Performance improvement: **38% and 44%** for **SpMV** and **SpMM** over the widely used CSR format
- SMASH is **highly efficient, low-cost** and **widely applicable**

SMASH

Co-Designing Software Compression and Hardware-Accelerated Indexing for Efficient Sparse Matrix Operations

Konstantinos Kanellopoulos, Nandita Vijaykumar, Christina Giannoula, Roknoddin Azizi, Skanda Koppula, Nika Mansouri Ghiasi, Taha Shahroodi, Juan Gomez Luna, Onur Mutlu

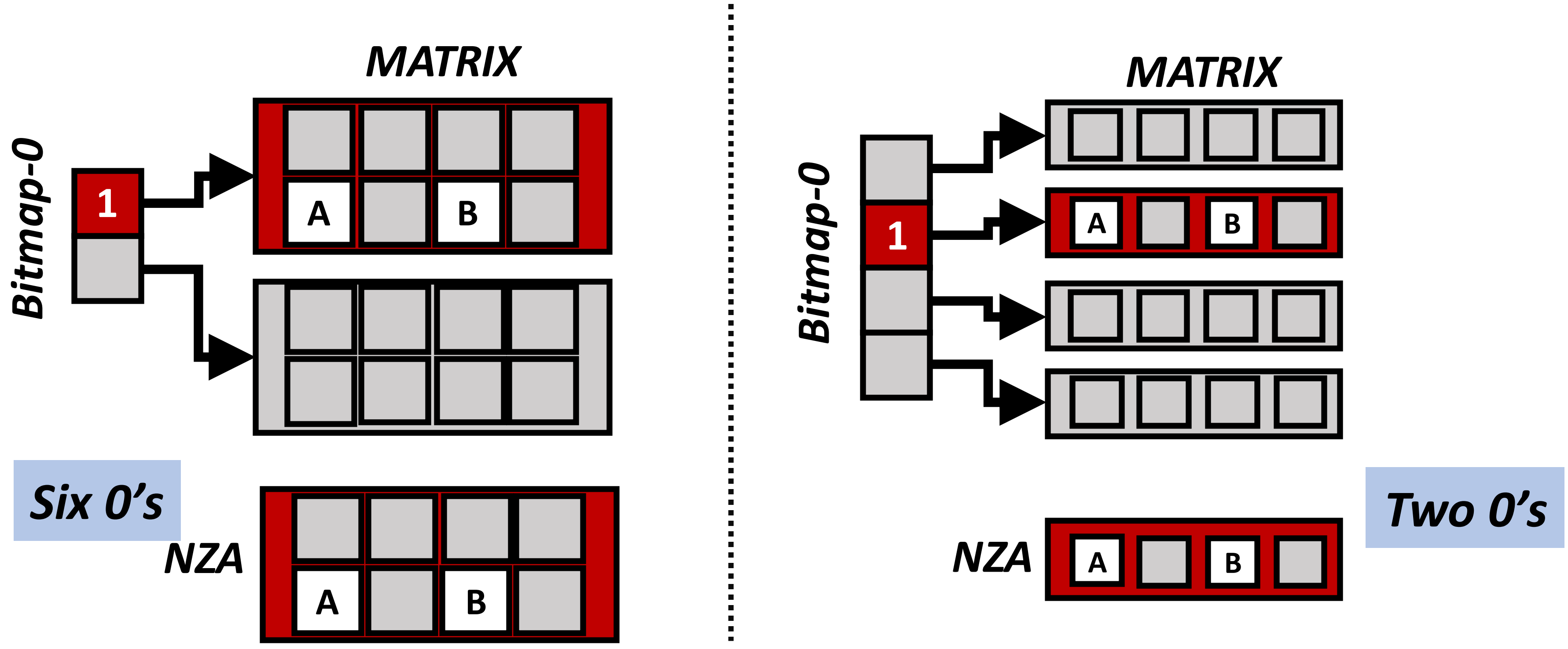
SAFARI

ETH zürich

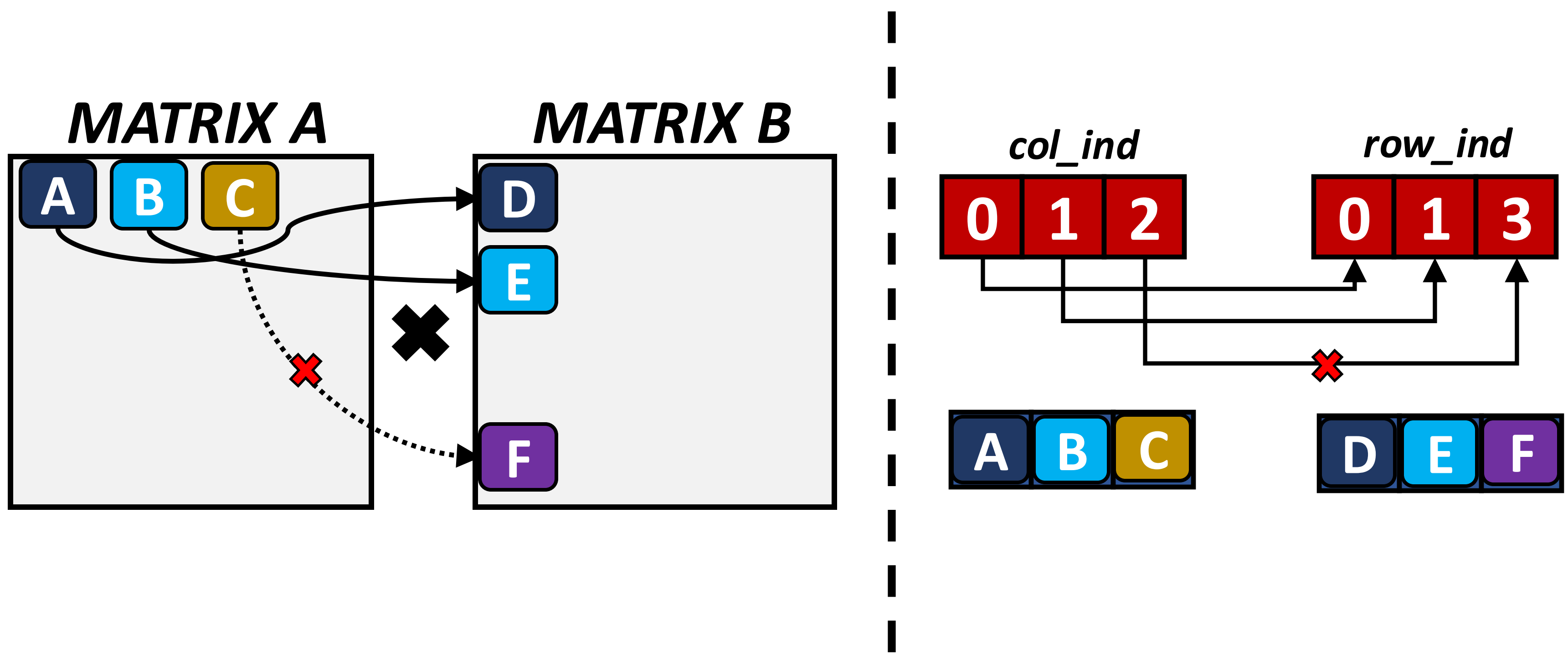
Carnegie Mellon University

Compression Ratio

SMALL BITMAPS VS ZERO COMPUTATION

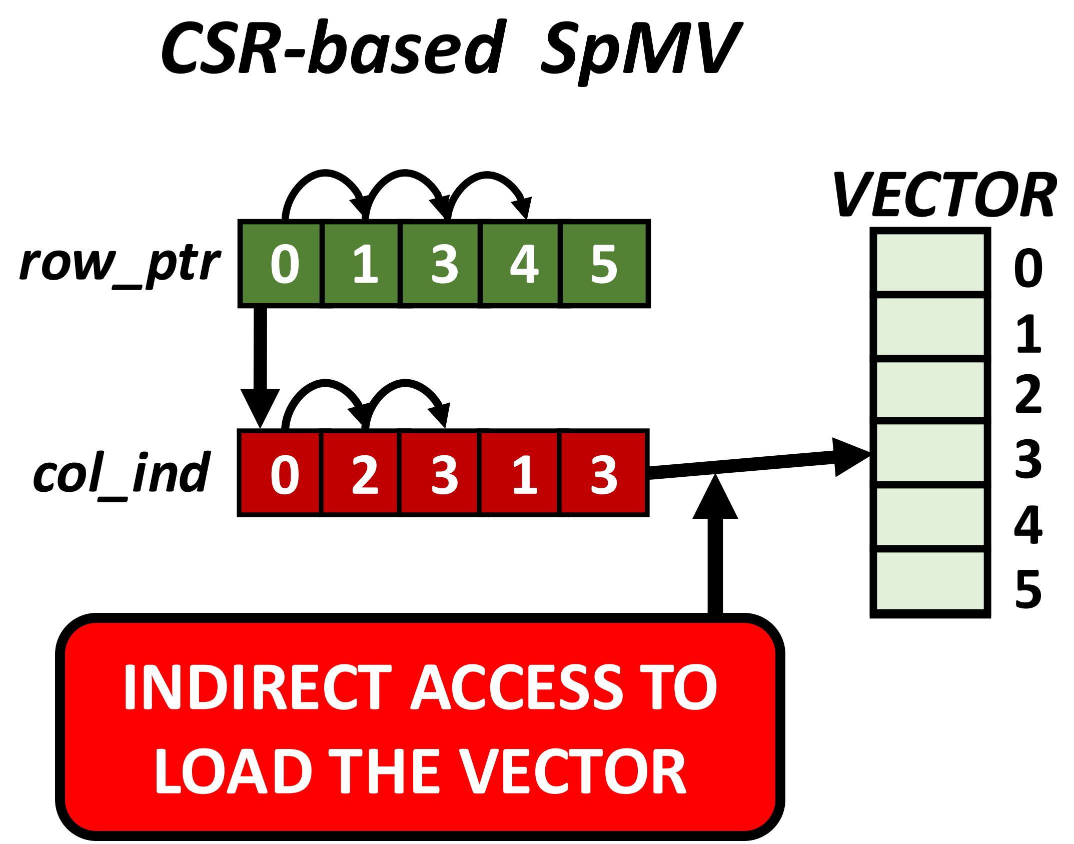


Sparse Matrix Matrix Multiplication with CSR

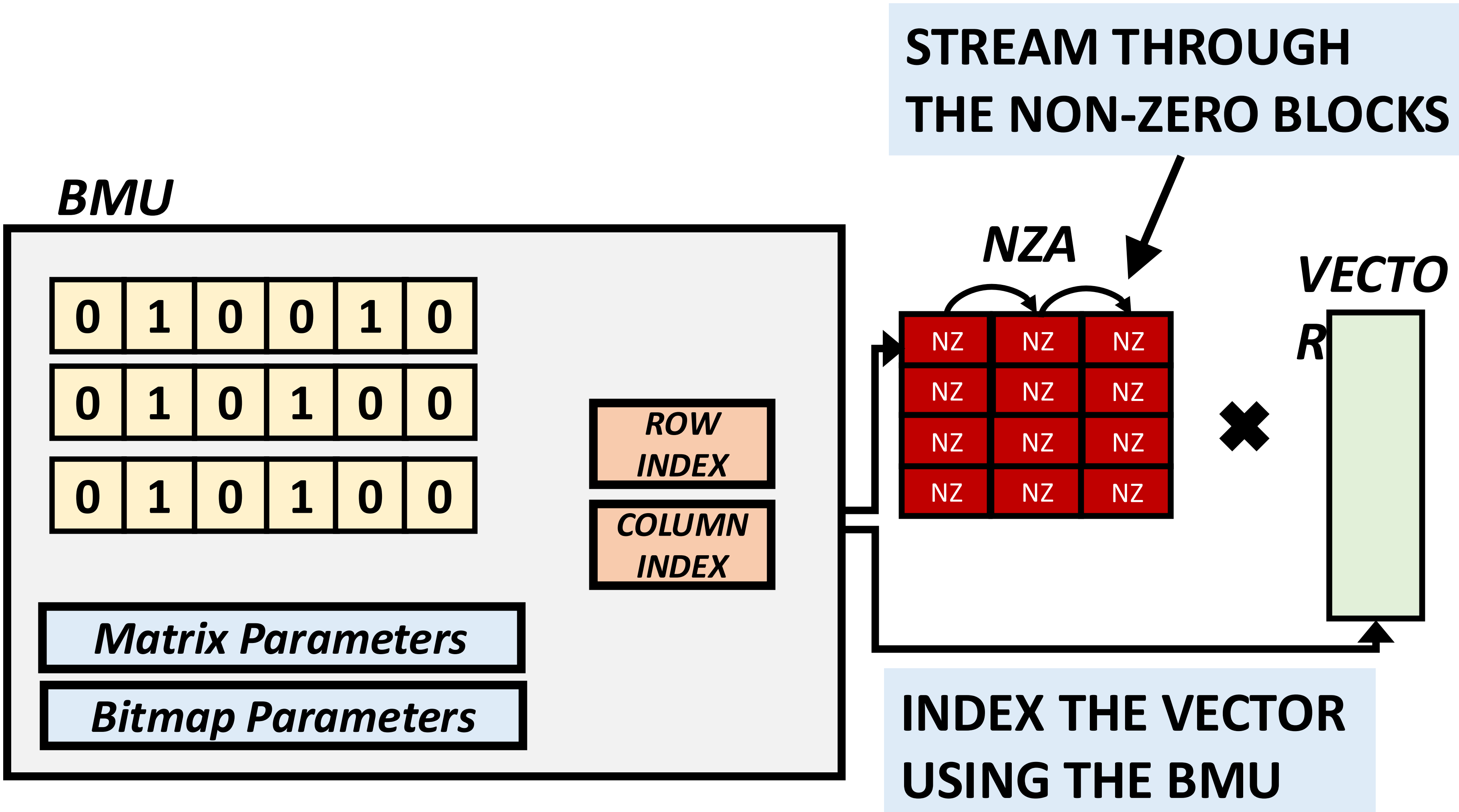


INDEX MATCHING FOR EVERY INNER PRODUCT

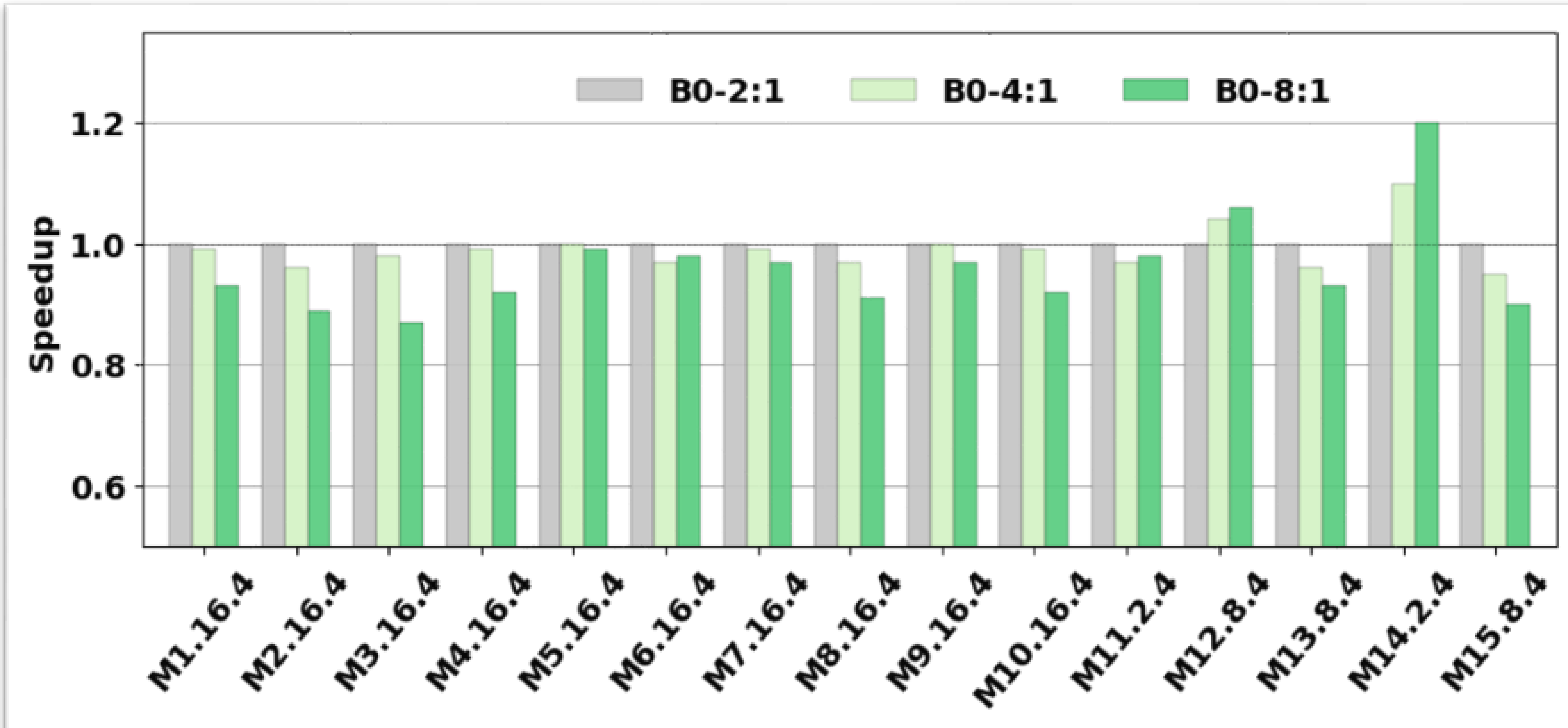
CSR-based SpMV



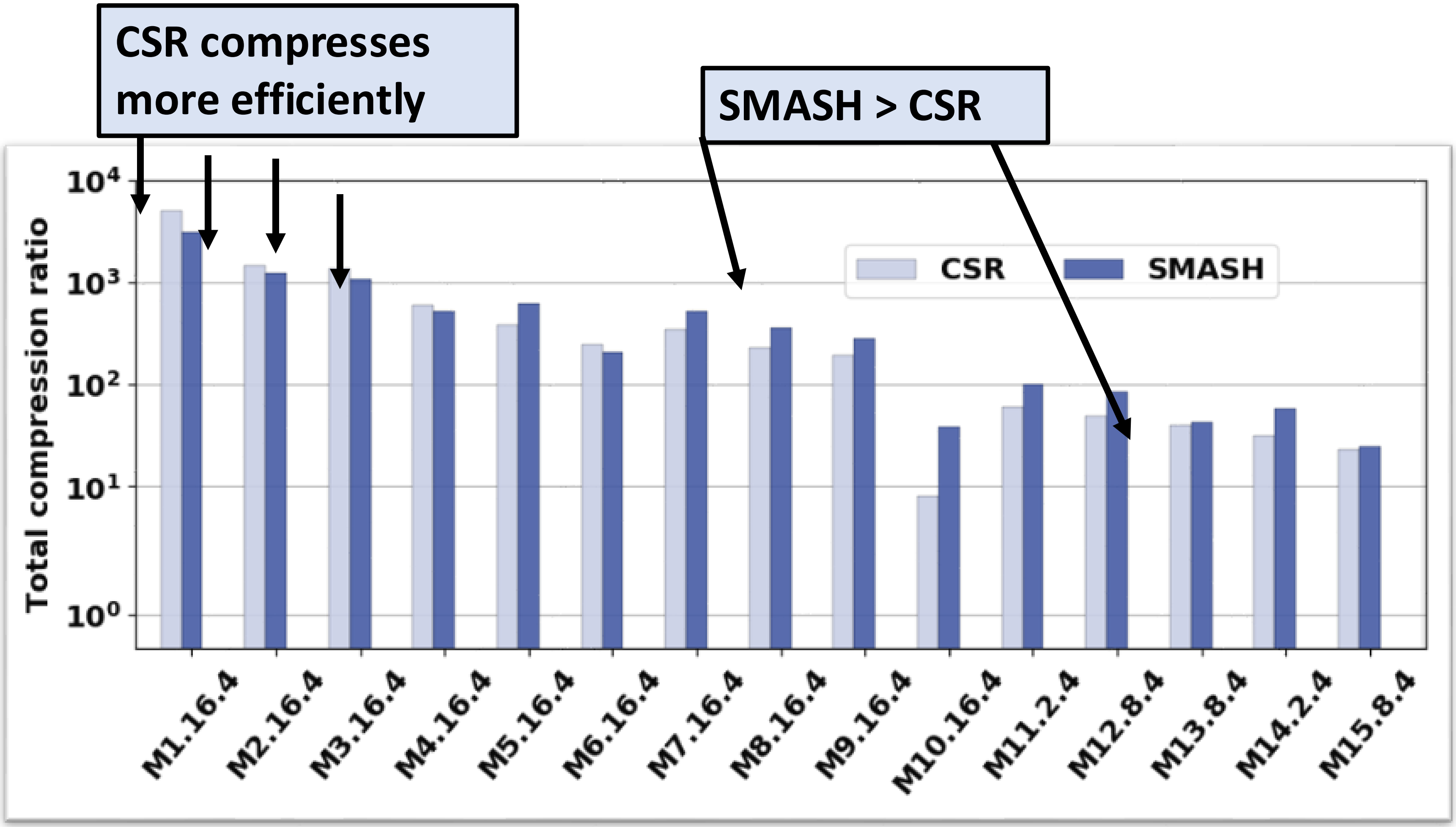
Use Case: SpMV



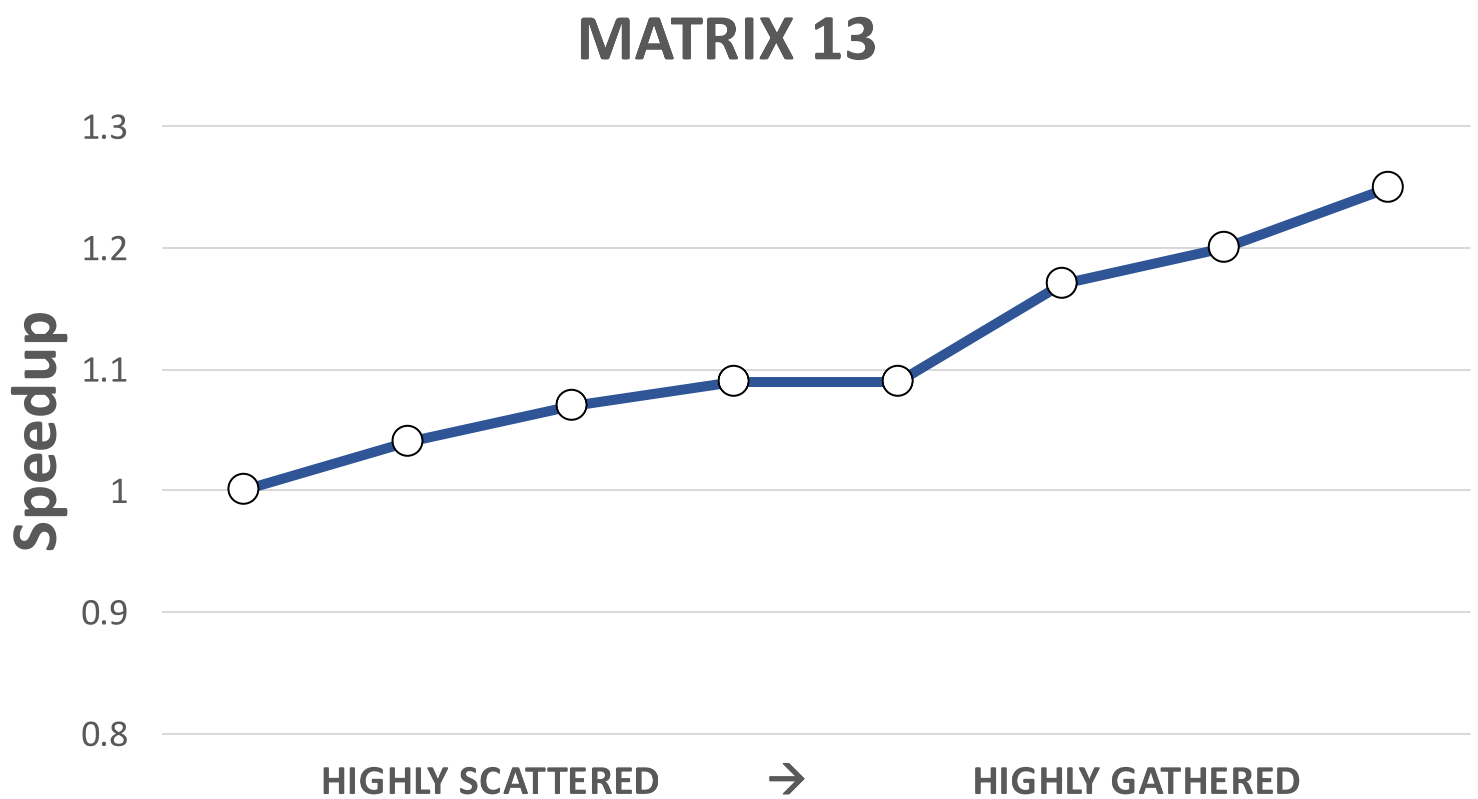
Effect of Compression Ratio



Storage Efficiency



Distribution of non-zero elements



Storage Efficiency