Cerberus: Triple Mode Acceleration of Sparse Matrix and Vector Multiplication

Soojin Hwang, Daehyeon Baek, Jongse Park, and Jaehyuk Huh

ACM TACO, 2024

School of Computing, KAIST

CASYS

Background & Motivation

Sparse Matrix

- Sparse Matrix
  - A matrix with a significant amount of zero values
  - Density = (# of non-zero elements)/(# of elements)
  - Sparse matrix: Density ≤ 0.5
  - Appears in various problem domains
  - Irregular/nondeterministic access pattern
  - Inefficient with dense matrix acceleration solutions

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

Dense Matrix

0	0	a	0
0	b	0	0
0	0	0	c
0	d	0	0

Sparse Matrix
(Density = 0.25)

Sparse Matrix

- Sparse Matrix
  - A matrix with a significant amount of zero values
  - Density = (# of non-zero elements)/(# of elements)
  - Sparse matrix: Density ≤ 0.5
  - Appears in various problem domains
  - Irregular/nondeterministic access pattern
  - Inefficient with dense matrix acceleration solutions
- Sparse Matrix and Vector Multiplication
  - SpMSpV: Sparse matrix – sparse vector
  - SpMV: Sparse matrix – dense vector
  - Building block of various workloads

Sparse Matrix and Vector Multiplication

- Building Block of Various Workloads
  - SpMSpV: Sparse matrix – sparse vector
  - SpMV: Sparse matrix – dense vector
- High-Performance Computing (HPC)
  - Graphs, big-data, linear programming, ...
  - Contains iteration of SpMVs
  - Vector sparsity can vary during execution
- Sparse Neural Networks

Sparse Neural Networks

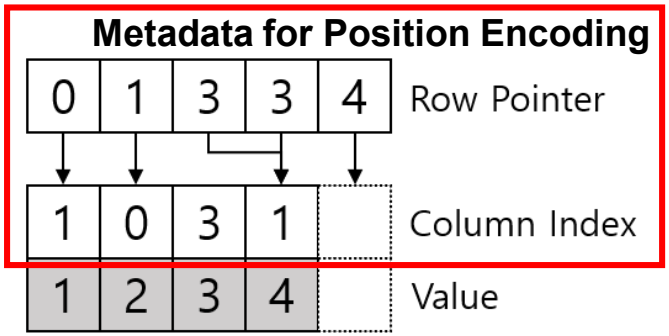
- Weight Sparsity (Matrix)
  - Appears by compression (e.g. Pruning)
  - Depends on the layer parameters
  - Static sparsity (does not change during inference)
  - Different level of sparsity for distinct layers
- Activation Sparsity (Vector)
  - Appears by activation functions (e.g. ReLU)
  - Depends on the user input
  - Dynamic sparsity: Co-existence of SpMSpV, SpMV
- Can Contain Dense Layers

Sparse Matrix Representation

- Dense Format
  - Saves all data in a single array
- Sparse Format
  - Saves only non-zero values
  - Encodes the position of non-zero values
  - Classified with metadata formats

0	1	0	0
2	0	0	3
0	0	0	0
0	4	0	0

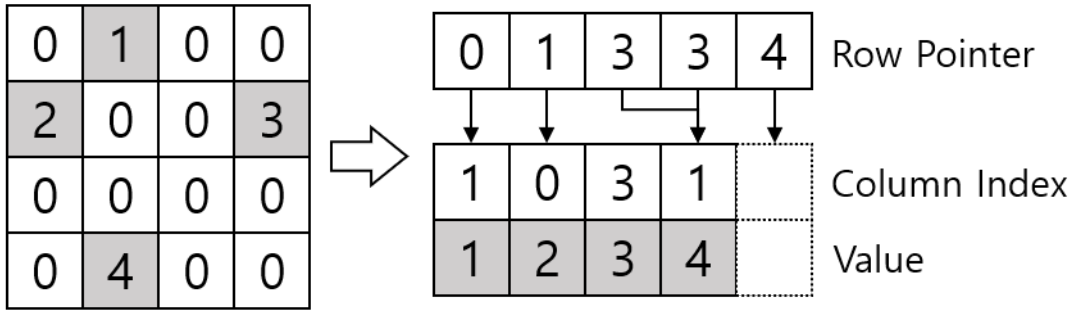
Dense Format



Sparse Format Example:
Compressed Sparse Row (CSR)

Sparse Matrix Representation

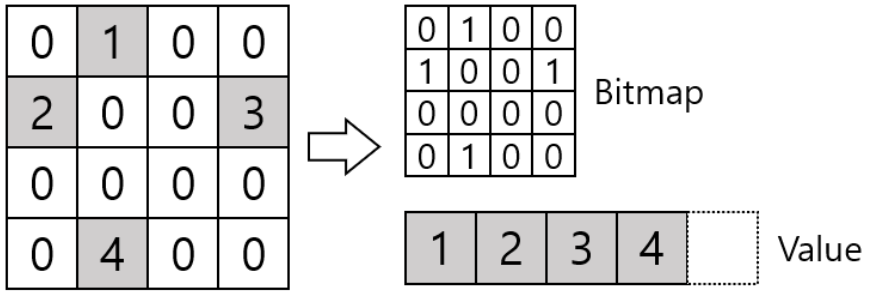
- Compressed Format
  - Encoding with pointer/index array
  - Compressed sparse row/column
  - Metadata size is proportional to # of non-zeros (nnz)



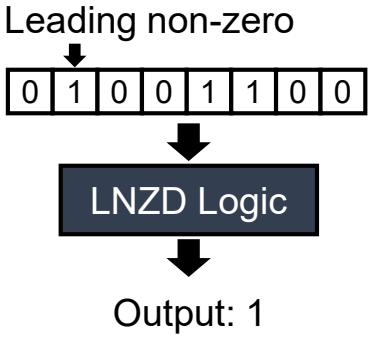
Row-major Compressed Format:
Compressed Sparse Row (CSR)

Sparse Matrix Representation

- Bitmap Format
  - Encoding with bitmap
  - Row/Column-major bitmap format
  - Requires leading non-zero detection (LNZD)
  - Metadata size is proportional to the matrix size



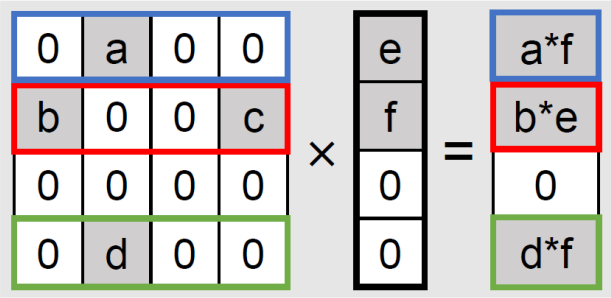
Row-Major Bitmap



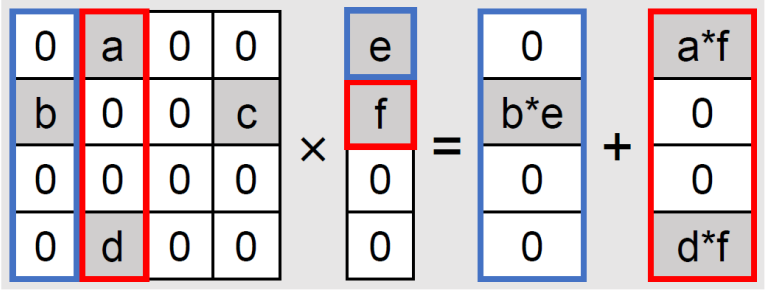
LNZD Operation

SpMV Algorithm

- Dot Product
  - General algorithm for dense MV
  - Dot product between matrix row and whole vector
  - Requires repetitive load of the input vector
- Scalar Multiplication
  - Multiply matrix column and vector element
  - Partial output vectors should be accumulated

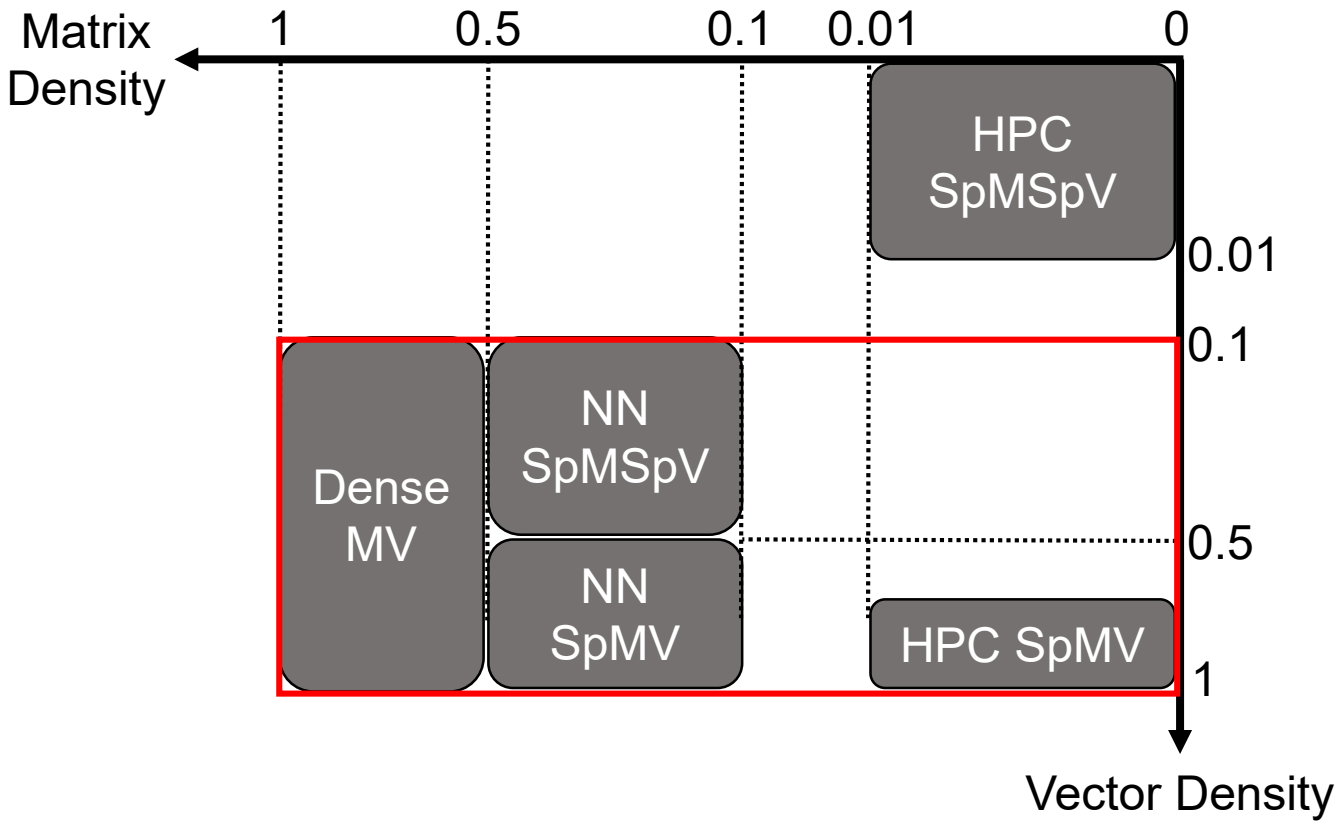


Dot Product

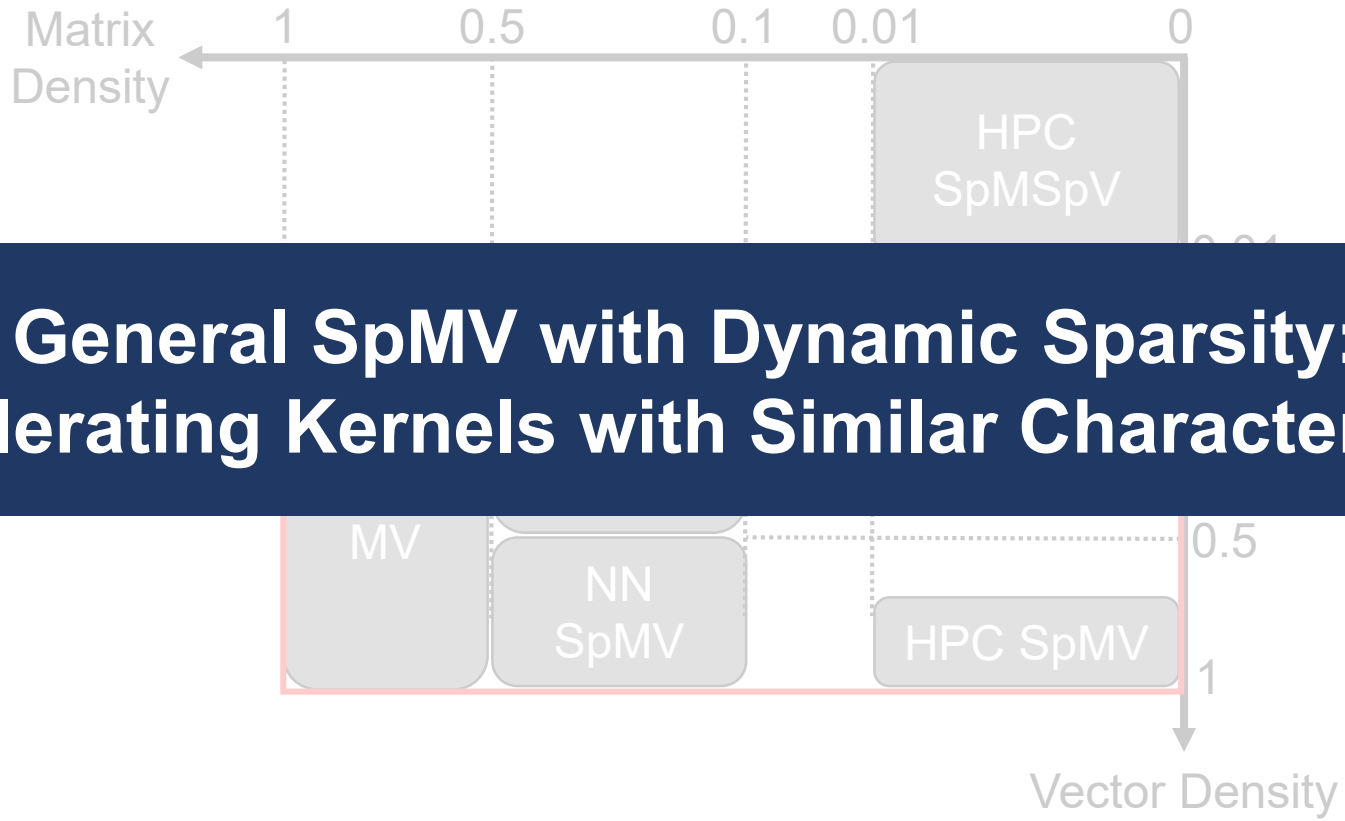


Scalar Multiplication

Problem Scope



Problem Scope



General SpMV with Dynamic Sparsity: Accelerating Kernels with Similar Characteristics

Prior Work

Name	Type	Target Workloads	Design Choices	
			Mtx. Format	Algorithm
NVIDIA cuSPARSE	SW Framework	General SpMV	Compressed	Dot Product
Intel MKL		General SpMV	Compressed	Dot Product
EIE (ISCA'16)	ASIC Accelerator	Sparse NN SpMSpV/SpMV	Compressed	Scalar Mult.
MASR (PACT'19)		Sparse NN SpMSpV	Bitmap	Dot Product
SIGMA (HPCA'20)		Sparse NN Nonsquare SpGEMM	Bitmap	Dot Product
Tensaurus (HPCA'20)		General Sparse-Dense Tensor Mult.	Compressed (CSR Variant)	Dot Product
SpaceA (HPCA'21)	PIM	HPC SpMV	Compressed	Dot Product

Prior Work

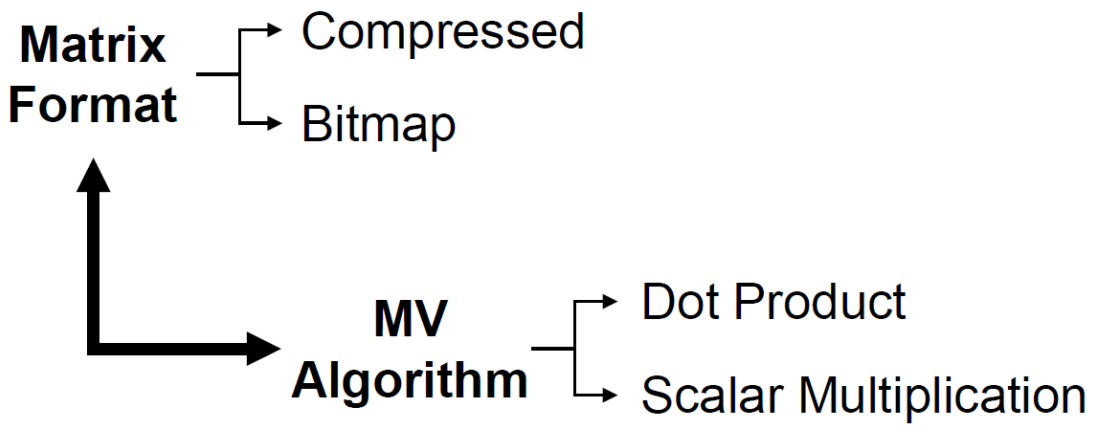
Name	Type	Target Workloads	Design Choices	
			Mtx. Format	Algorithm
NVIDIA cuSPARSE	SW Framework	General SpMV	Compressed	Dot Product
Intel MKL		General SpMV	Compressed	Dot Product
EIE (ISCA'16)	ASIC	Sparse NN SpMSpV/SpMV	Compressed	Scalar Mult.
MASR (PACT'19)		Sparse NN SpMSpV	Bitmap	Dot Product
(HPCA'21)	PIM	HPC SpMV	Compressed	Dot Product

Limitation:
Various Design Choices,
No Quantitative Analysis

Design Space Exploration

SpMV Design Space Definition

- Two axes
  - Matrix format
  - SpMV algorithm



SpMV Design Space Definition

- Two axes
  - Matrix format
  - SpMV algorithm
- Four Design Choices (Accelerator Classes)

Matrix Format	MV Algorithm	Acronym
Compressed	Dot Product	C_mD_a
	Scalar Mult.	C_mS_a
Bitmap	Dot Product	B_mD_a
	Scalar Mult.	B_mS_a

Methodology

- 18 Real-world SpMV benchmarks

Sparse Neural Network Kernels

Name	M	N	d_m	d_v
AlexNet-FC1	4096	9216	0.09	0.30
AlexNet-FC2	4096	4096	0.09	0.38
AlexNet-FC3	1000	4096	0.25	0.46
BERT-Pooler	768	768	0.20	1.0
GPT2-FC	50257	768	0.13	1.0
LSTM-Input1	6000	1500	0.3	0.3
LSTM-Input2	6000	1500	0.3	1.0
MobileNet-FC	1000	1280	0.28	0.66
ResNet-FC	1000	2048	0.31	0.83
Selfish-RHN1	1660	830	0.15	1.0

HPC Workloads

Name	Problem Domain	M	N	d_m
bcsstk10	Structural Prob.	1086	1086	0.02
bcsstk13	Computational Fluid Dynamics	2003	2003	0.02
bcsstk17	Structural Prob.	10974	10974	0.004
c8_mat11	Combinatorial Prob.	4562	5761	0.09
kl02	Linear Programming Prob.	71	36699	0.08
lhr34c	Chemical Process Simulation	35152	35152	6×10^{-4}
pdb1HYS	Weighted Unidirected Graph	36417	36417	0.002
psmigr_1	Economic Prob.	3140	3140	0.06

Methodology

- 600 Microbenchmarks
- Cycle-accurate C++ Simulator

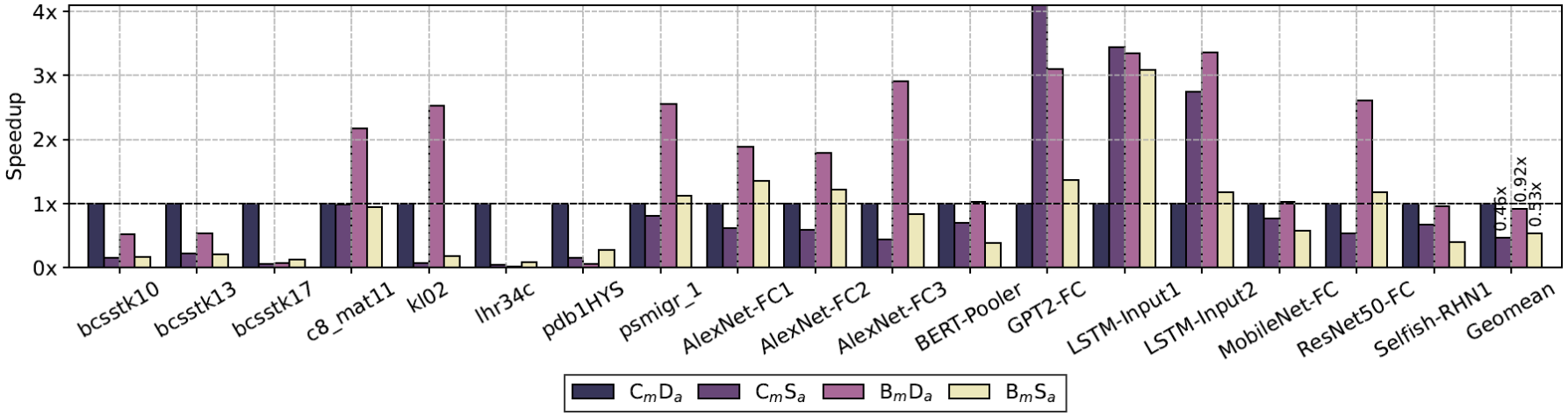
Hyperparameters	Values
M	512, 1024, 2048, 4096
N	512, 1024, 2048, 4096, 8192, 16384
d_m	0.01, 0.05, 0.1, 0.2, 0.3
d_v	0.2, 0.4, 0.6, 0.8, 1.0

PE Configurations	Param.
PE Array Configuration	256×1
Bitmap Register Capacity	64B
Integer Mult. Latency	1 cycle
Integer Add. Latency	1 cycle
LNZD Latency (32-bit window)	1 cycle
Value Precision	INT16

Memory Configurations	Param.
# of SPM Ports	4
SPM Capacity per PE	16KB
SPM Access Latency	1ns
Off-chip DRAM Bandwidth	600GB/s
Off-chip Access Latency	100ns
Chip Frequency	1GHz

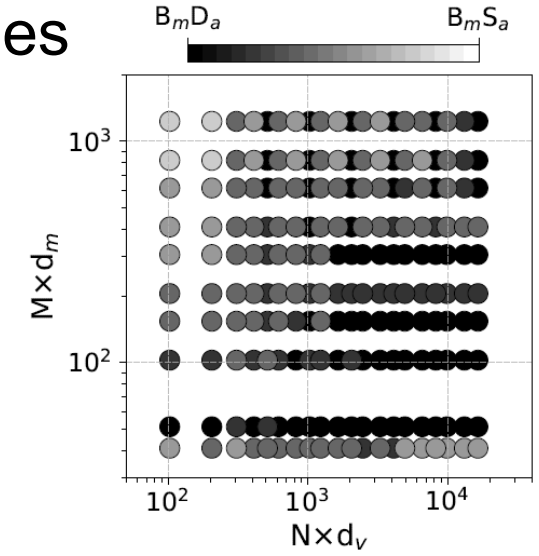
Overall Performance Trend of Classes

- Speedup with Real-world Benchmarks
  - Normalized to the best-performing accelerator class
  - Average best performance (Single-best): C_mD_a
- Single-best (C_mD_a) is not Always Best
  - Other classes outperform in more than half cases



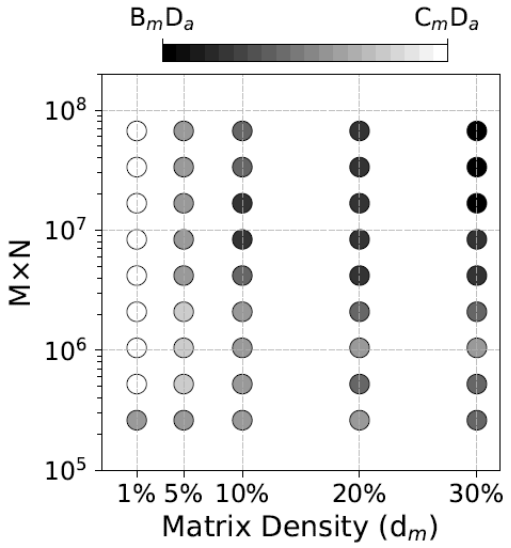
Performance Tradeoff: MV Algorithm

- Microbenchmark Experiment
- Condition for Scalar Multiplication Preference:
  - Large number of non-zeros in vector
  - Large number of non-zeros per matrix column
- Meaning of Two Conditions:
  - Input vector load overhead increases
  - Incurs slowdown of dot product



Performance Tradeoff: Matrix Format

- Microbenchmark Experiment
- Condition of Compressed > Bitmap:
  - Low matrix density
  - Number of matrix element is high (i.e. Large Matrix)
- Impact of Low Matrix Density:
  - CSR storage overhead decreases
- Impact of Large Matrix Size:
  - Bitmap LNZD overhead increases



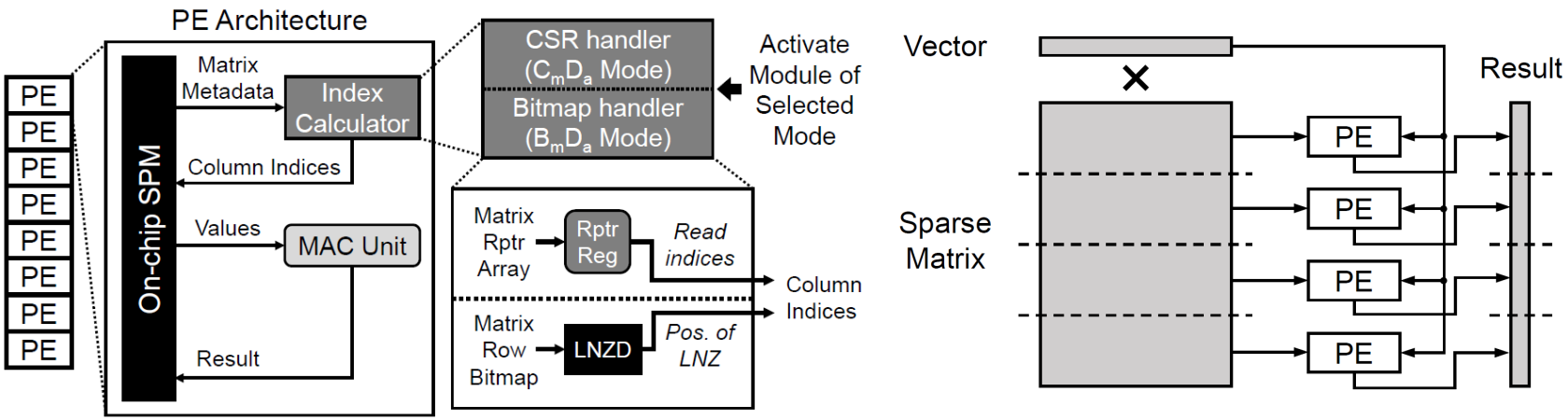
Potential of Multi-mode Acceleration

- No Single-best Design for All Workloads
- Input Characteristic Determines Preference
  - Matrix size (# of rows, columns), Matrix density
- Solution: Combining Multiple Design Choices
  - Requirement: Low hardware overhead
- Three Modes
  - C_mD_a: Single-best accelerator class
  - B_mD_a: Same dataflow, complementary perf.
  - D_mD_a: Dense MV mode

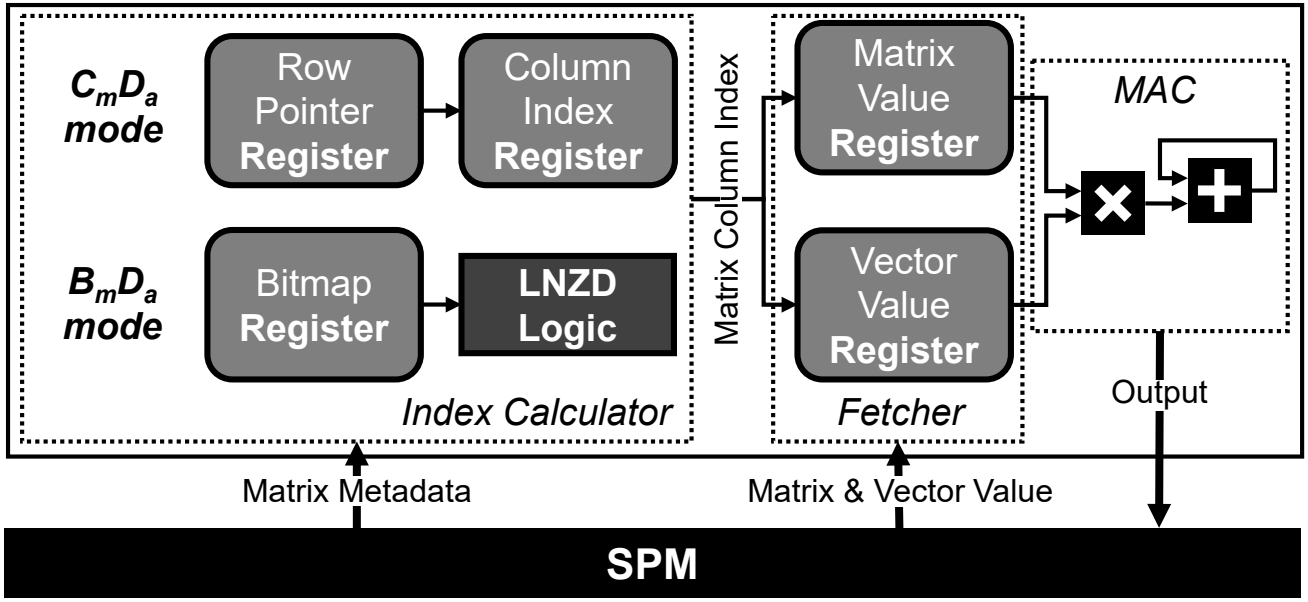
Design: Cerberus

Overview of Cerberus

- Triple-mode SpMV Accelerator
  - Heterogeneity limited in Index Calculator
  - Minimize hardware overheads from multiple modes
- 1D PE Array Suitable for MV Kernels
- Select Mode using Software Selector

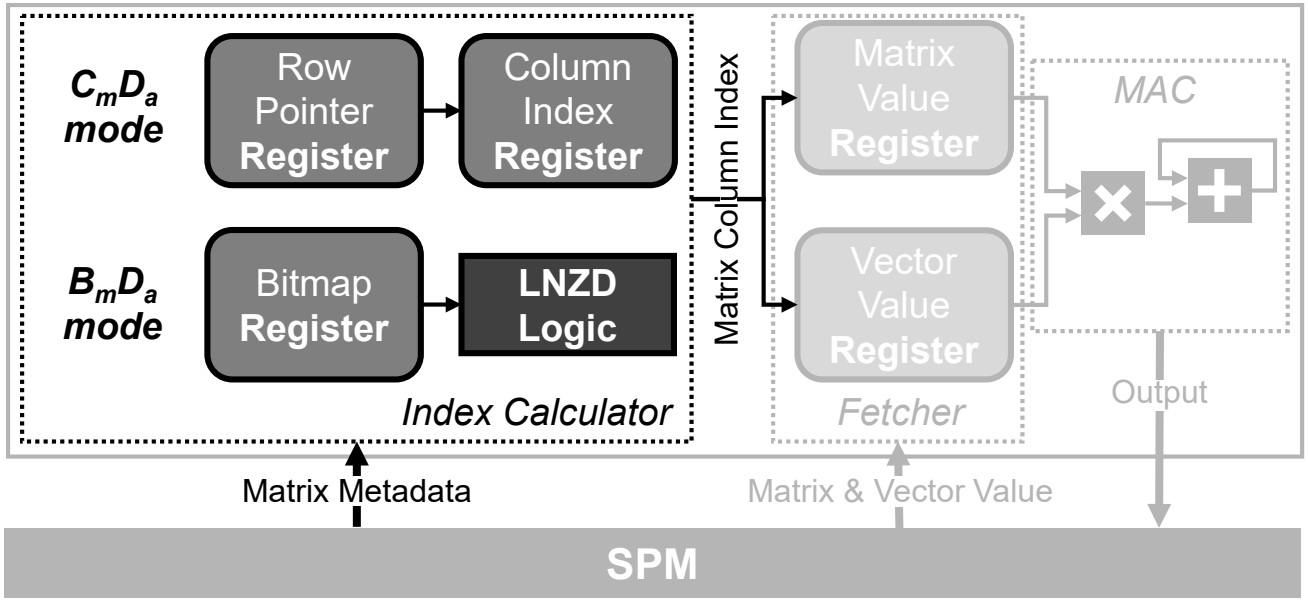


Design: PE Architecture



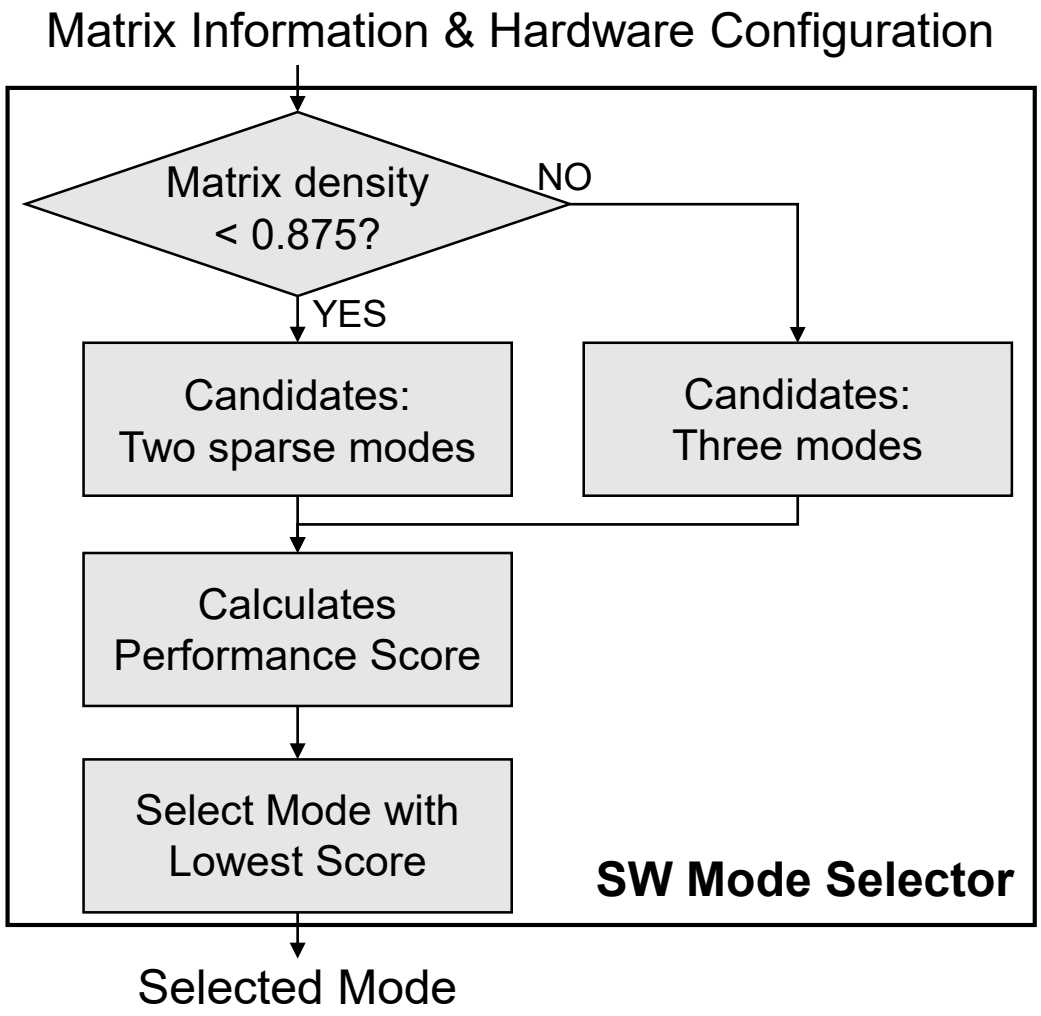
- Processing Element (PE) of Cerberus
  - On-chip SPM: Saves input and output
  - Fetcher: Fetches dot product operands
  - MAC: Performs dot product

Design: PE Architecture

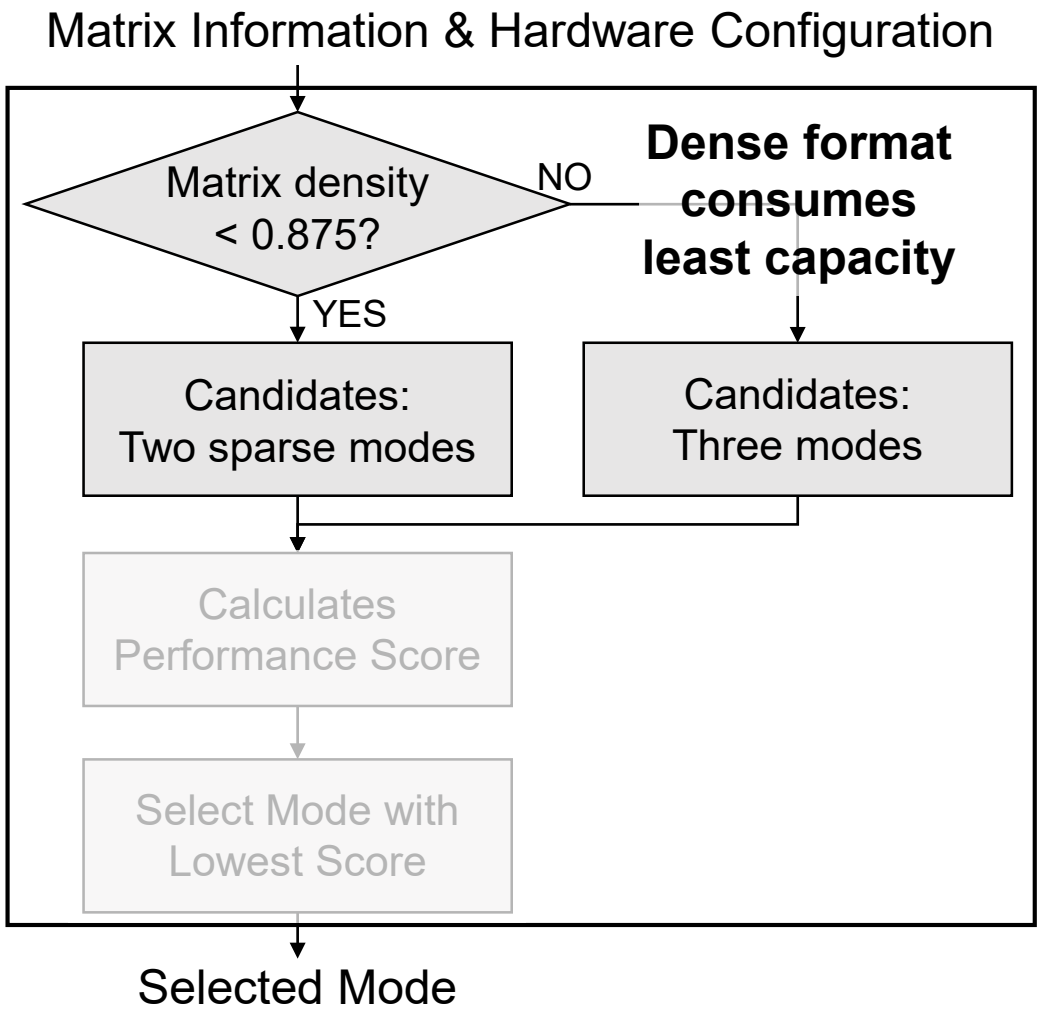


- Index Calculator
  - Calculates the exact position of non-zero values
  - Two modules for two sparse modes
  - Inactivates unused module(s) runtime

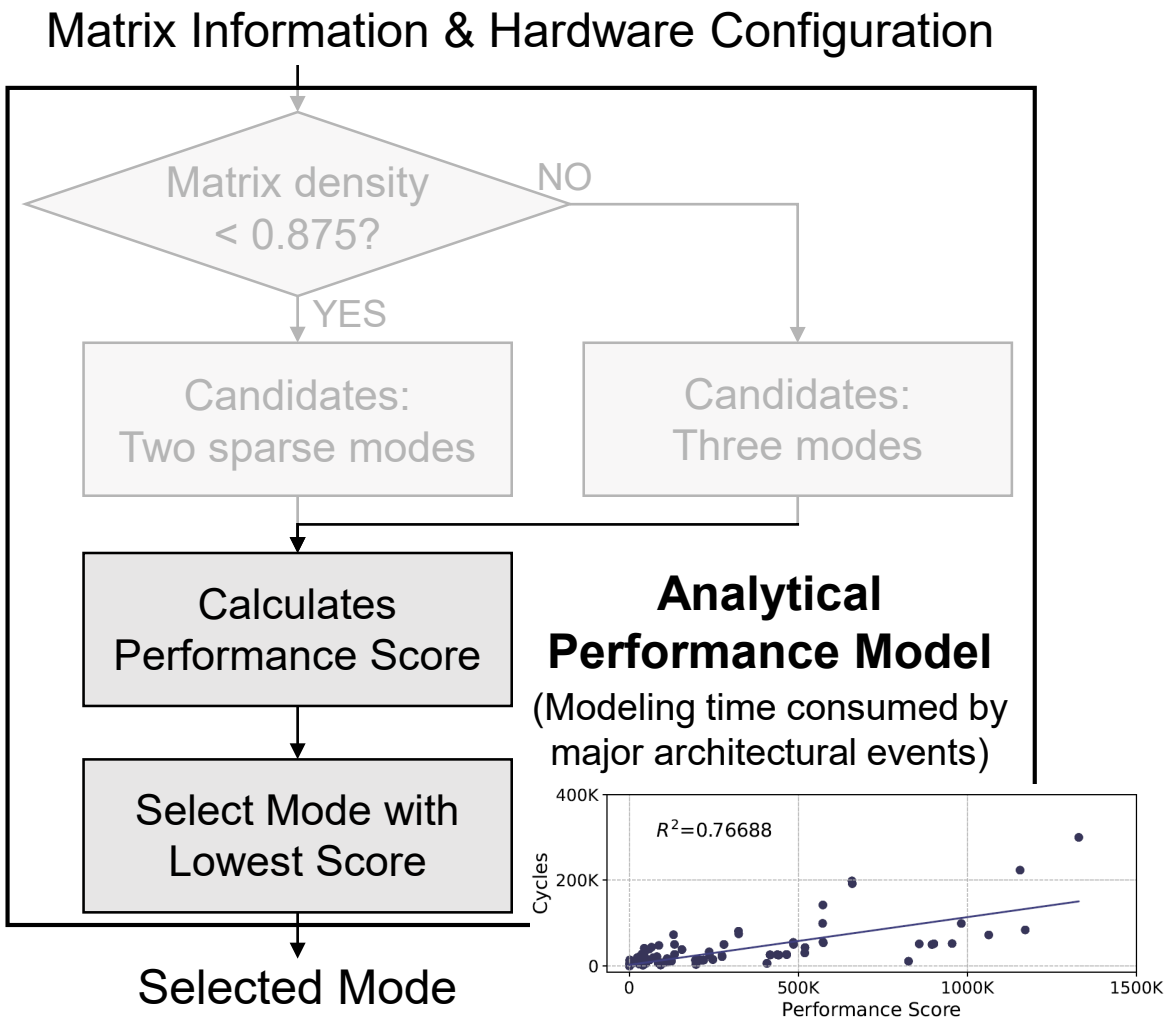
Design: Mode Selector



Design: Mode Selector



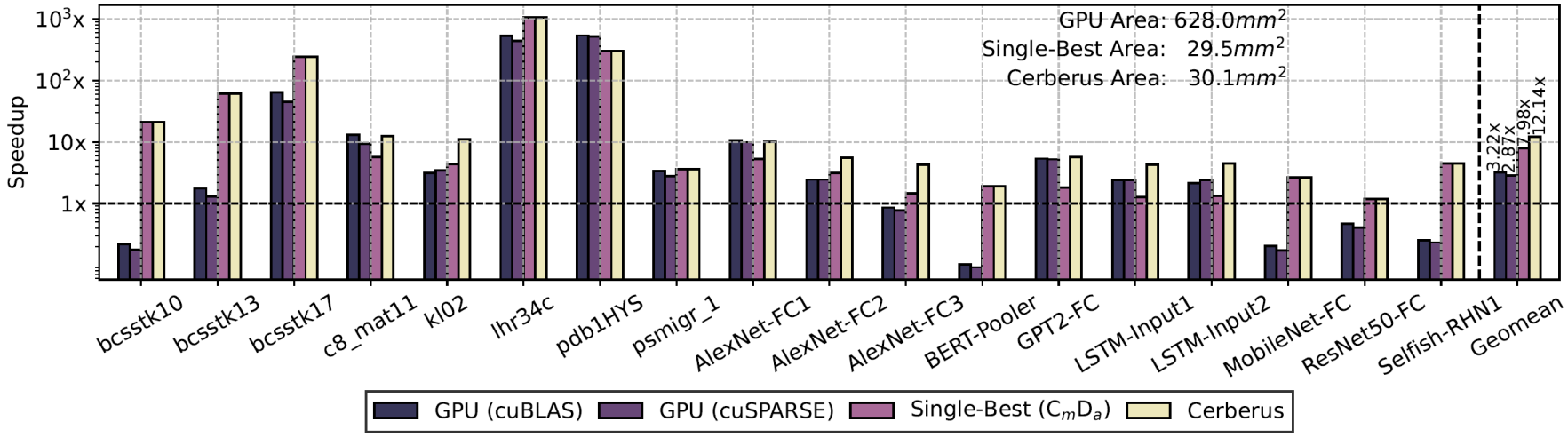
Design: Mode Selector



Evaluation

Real-World Benchmark Speedup

- Average Speedup:
  - vs. Baseline (Dense MV Accelerator): 12.1×
  - vs. Dense GPU (cuBLAS): 3.8×
  - vs. Sparse GPU (cuSPARSE): 4.2×
  - vs. Single-best ($C_mD_vD_a$ -only): 1.5×



Case Study: Sparse Neural Networks

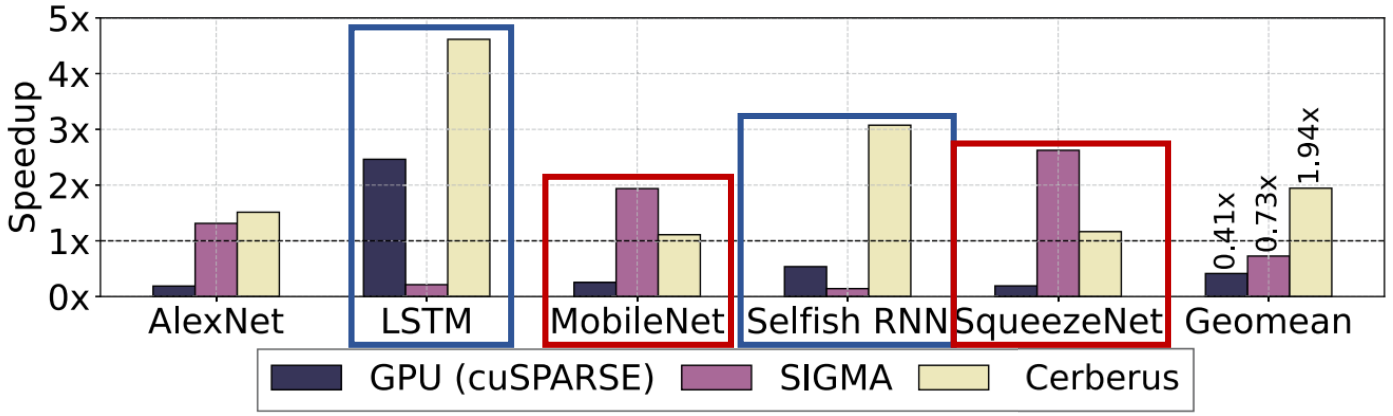
- Five Sparse Neural Network Benchmarks
  - Excluded activation functions
  - Converted Conv into multiple SpMV kernels
- Comparison with SIGMA, GPU
  - SIGMA[1)]: Evaluated with STONNE simulator
  - GPU: cuSPARSE, SpGEMM/SpMV Routines

Model	Layers	Avg. Weight Density	Avg. Act. Density
AlexNet	5 Conv, 3 FC	0.11	0.59
LSTM	2 LSTM	0.3	0.82
MobileNet	14 Conv, 1 FC	0.48	0.60
Selfish RNN	10 RNH, 1 FC	0.47	0.99
SqueezeNet	2 Conv, 8 Fire block	0.33	0.78

1) Qin et al., “SIGMA: A Sparse and Irregular GEMM Accelerator with Flexible Interconnects for DNN Training”, HPCA 2020

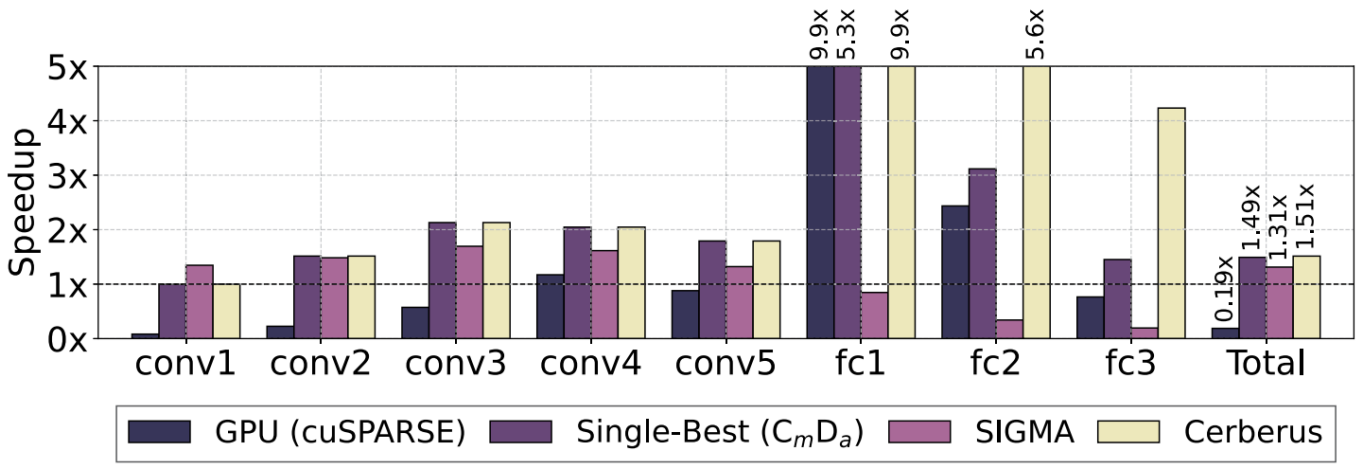
Case Study: Sparse Neural Networks

- vs. Baseline: 1.9× on Average
- vs. Sparse GPU: 4.7× on Average
- vs. SIGMA: 2.7× on Average
  - Cerberus performs better with MV-major models
  - SIGMA performs better with Conv-major models
  - However, Cerberus can adapt in various situations



Case Study: Sparse AlexNet

- Layer-wise Breakdown
- Various Layer Selection
  - Selects C_mD_a mode for 4 Conv layers
  - Selects D_mD_a mode for 1 Conv layer
  - Selects B_mD_a mode for 3 FC layers
  - Shows the reliability of triple-mode design



Hardware Overhead

- Estimate Overhead of Triple-mode Support
  - Compared to the single-best SpMV accelerator
  - Relatively small overhead

Components	Area (mm^2)		Power (mW)	
	Single-best	Cerberus	Single-best	Cerberus
SPM	29.2	29.2	13978.6	13978.6
LNZD Logic	-	0.055	-	0.61
Registers	0.12	0.73	505.8	3034.6
Adders	0.052	0.052	2.13	2.13
Multipliers	0.15	0.15	16.70	16.70
Total	29.5	30.1 **(+2.2%)**	14503.2	17032.7 **(+17.4%)**

More Results in Paper

- Sensitivity Study
- Microbenchmark Evaluation
- Data Partitioning Scheme
- RTL Implementation

Summary

- No Single-best Design for All SpMVs
  - Best design is decided by input characteristics
- Design: Cerberus
  - Triple-mode SpMV Accelerator
  - Heterogeneity in computational units
  - Selects execution mode with SW selector
- Outperforms Single-best by 1.5×
  - Efficiently handles various layers in sparse NN
  - Negligible hardware overhead (+2.2% area)

Backup Slides

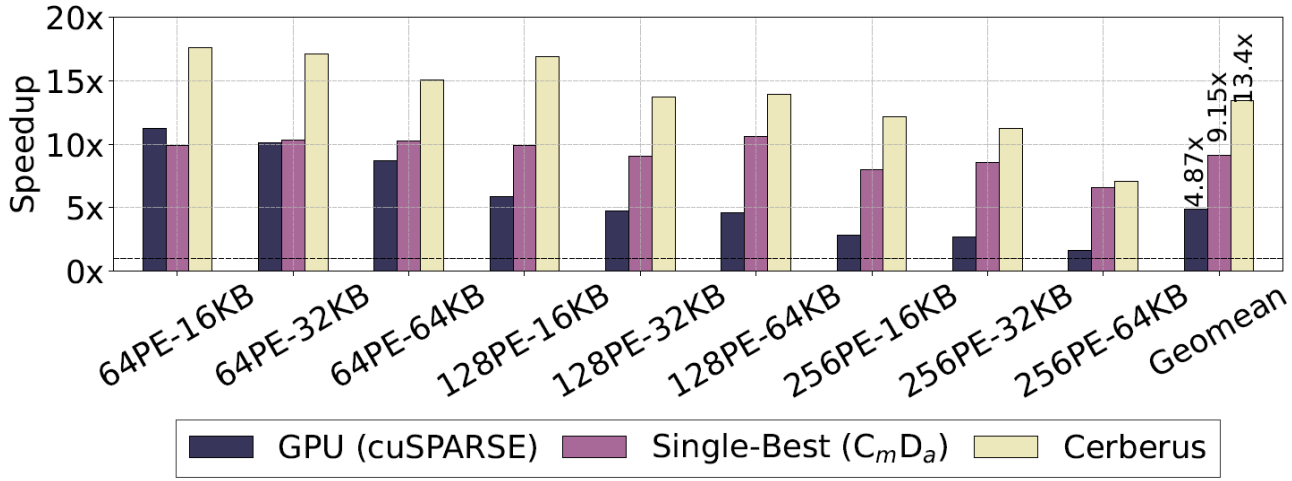
Performance Model Details

$$score = \frac{\left\{\max\left(\frac{total\_onchip\_access\_time(mode)}{\#\ of\ spm\ ports}, total\_additional\_ops\_time(mode)\right) + total\_offchip\_access\_time(mode)\right\}}{\#\ of\ processing\ elements}$$

Class	On-chip Access	Additional Operations	Re-Access Window	Off-chip Access
D_mD_a	2×M×N/ value_size/ prefetching_unit + M	M×N - M×N×d_m (Multiplication)	spm_blocksize + 2×N×value_size	{(Re-access window) > spm_capacity} ? {(M×N + N×E + M×value_size)/spm_blocksize} : {(2×M×N×2 + M×value_size)/spm_blocksize}
C_mD_a	M + 1 + M×N×d_m× (index_size + value_size)/ prefetching_unit + M×N×d_m + M	None	2×spm_blocksize + N×d_m×(index_size + value_size) + value_size×N	{(Re-access window) > spm_capacity} ? {(ptr_size×(M + E) + M×N×d_m×(index_size + value_size) + N×E×val_size + M×value_size)/spm_blocksize} : {(ptr_size×(M + E) + M×N×d_m×(index_size + value_size) + M×N×value_size)/spm_blocksize}
B_mD_a	M×N/(bitmapreg_size×8) + M×N×d_m×value_size/ prefetching_unit + M×N×d_m + M	M×N×d_m (LNZD)	spm_blocksize + N/8 + N×d_m×value_size + N×value_size	{(Re-access window) > spm_capacity} ? {(M×N/8 + M×N×d_m×value_size + N×E×value_size + M×value_size)/spm_blocksize} {(M×N/8 + M×N×d_m×value_size + M×N×value_size)/spm_blocksize + M}

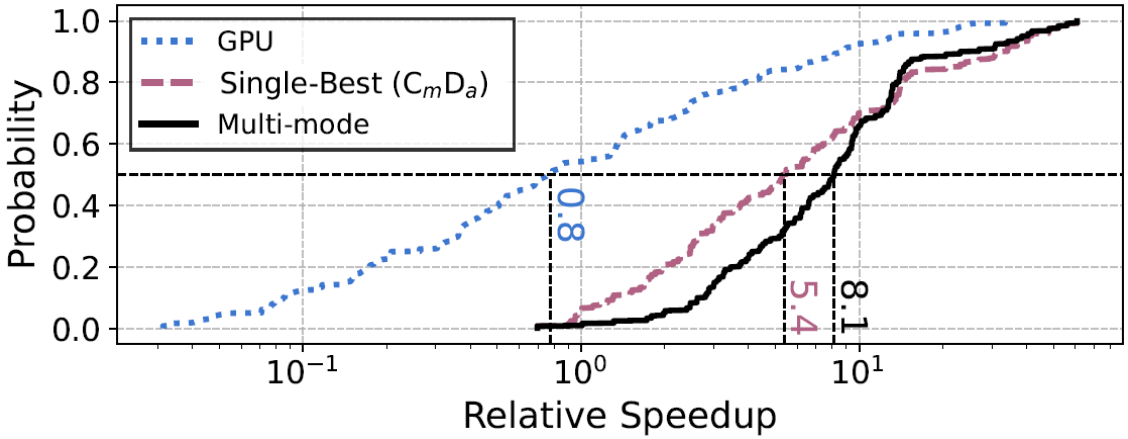
Sensitivity Study

- Testing Generality with Alternative HW Configs
  - Number of PEs
  - SPM capacity per PE
- Result: Cerberus is Robust to the Change of Hardware Configurations



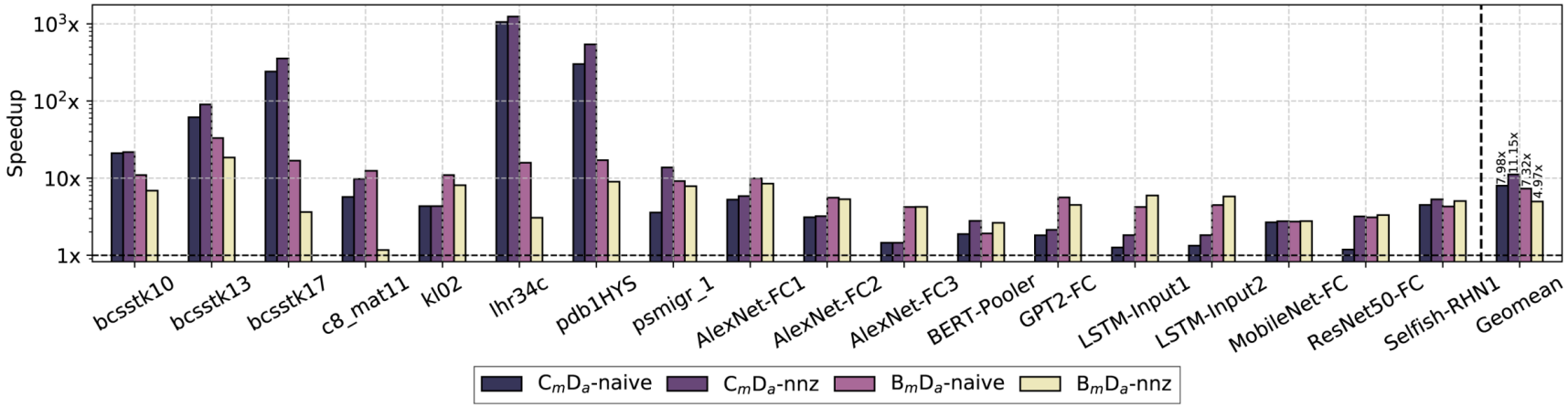
Microbenchmark Evaluation

- Cumulative Distribution Function (CDF)
- vs. Baseline: 8.1× in Median
  - Outperform baseline in general, while GPU doesn't
- vs. Single-best: 1.3× in Median
  - Shows necessity of second sparse mode



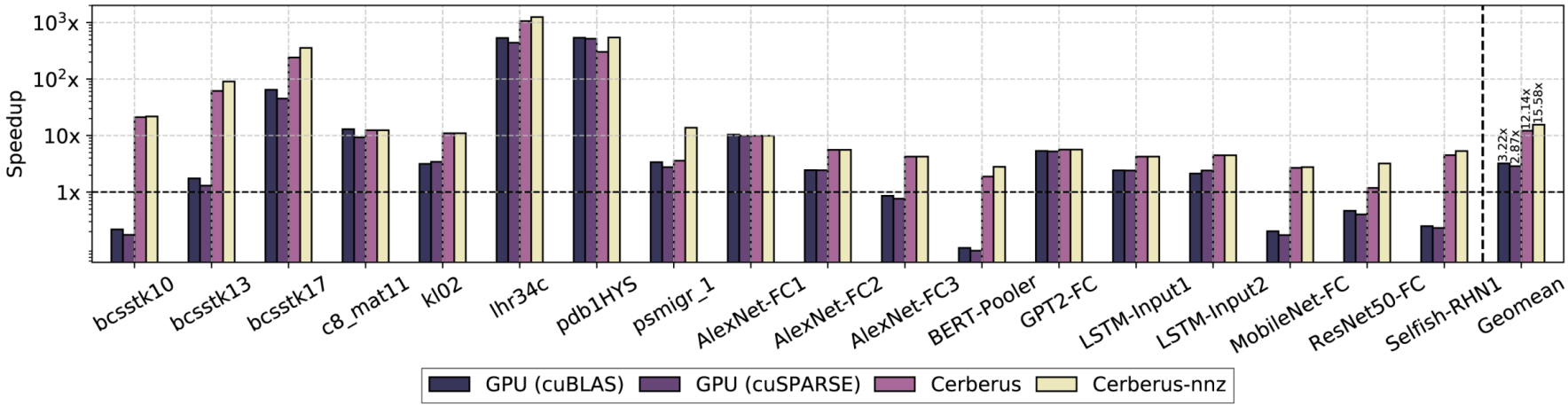
Partitioning Scheme (1)

- Naive Partitioning: Risk of Load Imbalance
  - Partition matrix with respect to the # of rows
- Alternative Scheme: NNZ-aware Partitioning
  - Partition matrix with respect to the # of non-zeros
  - Row-granularity
  - Different preference of two sparse modes



Partitioning Scheme (2)

- Applying Best-partitioning Scheme
  - NNZ-aware partitioning for $C_mD_vD_a$
- +28% Speedup over Naive Partitioning
  - Performance improvement of $C_mD_vD_a$
  - Revising misprediction of mode selector
    - Naive partitioning: 4 misprediction → 90% of oracle



RTL Implementation

- Modeled with Chisel, Compiled to Verilog
- Verified with Xilinx Vivado v2021.2
- On-chip Components
  - Behavioral simulation
  - Synthesis
  - Post-synthesis timing simulation
- Execution Flow of Three Modes
  - Behavioral simulation
  - Synthesis

Mode	Cells	I/O Ports	Nets
D_mD_a	114	68	1804
B_mD_a	261	100	3367
C_mD_a	316	100	3189