Accelerator Architectures for Machine Learning (AAML)

Lecture 9: Sparse DNN Accelerator

Tsung Tai Yeh
Department of Computer Science
National Yang-Ming Chiao Tung University

Acknowledgements and Disclaimer

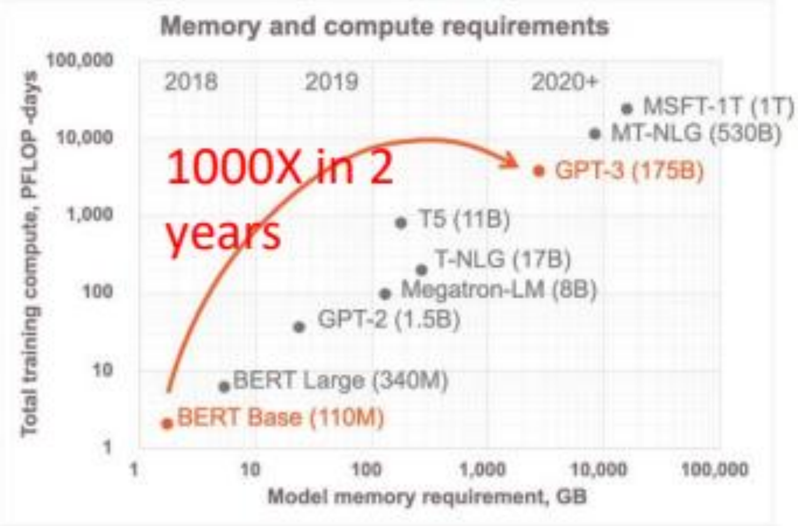
- Slides was developed in the reference with
  Joel Emer, Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, ISCA 2019 tutorial
  Efficient Processing of Deep Neural Network, Vivienne Sze, Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, Morgan and Claypool Publisher, 2020
  Yakun Sophia Shao, EE290-2: Hardware for Machine Learning, UC Berkeley, 2020
  CS231n Convolutional Neural Networks for Visual Recognition, Stanford University, 2020
  CS224W: Machine Learning with Graphs, Stanford University, 2021

Outline

- Sparse Transformer Model
- TPU v4 Sparse Core
- Nvidia Tensor Core: M:N Sparsity
- Cnvlutin Sparse Accelerator
- TorchSparse: Sparse CONV on the GPU

Unsustainable ML Model Growth

- We need a better way to grow models more efficiently
- Get the advantages of larger models but with substantially less compute and memory resources



Sparsity might be one of answer

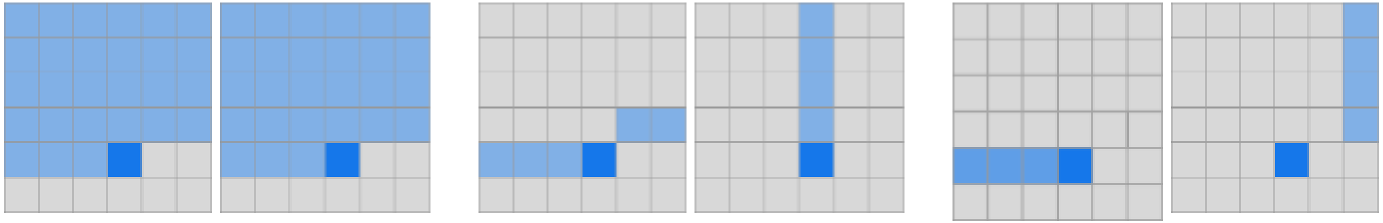
Deep Transformer Models

- Attention memory usage for a deep Transformer
  - 64 layers and 4 heads, recomputed during the backward pass
  - Requires the creation of an N x N attention matrix for every layer and attention head

Data type	Stored	Recomputed
1024 text tokens (several paragraphs)	1.0 GB	16 MB
32×32×3 pixels (CIFAR-10 image)	9.6 GB	151 MB
64×64×3 pixels (Imagenet 64 image)	154 GB	2.4 GB
24,000 samples (~2 seconds of 12 kHz audio)	590 GB	9.2GB

Sparse Transformer

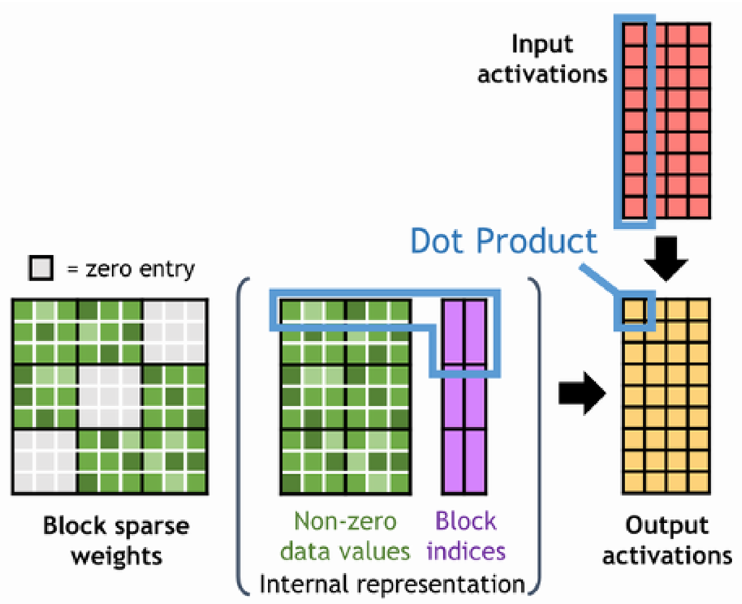
- Two-dimensional factorization of the attention matrix
  - The stride attention is roughly equivalent to each position attending to its row and its column
  - The fixed attention attends to a fixed column and the elements after the latest column elements



Normal transformer Strided attention Fixed attention

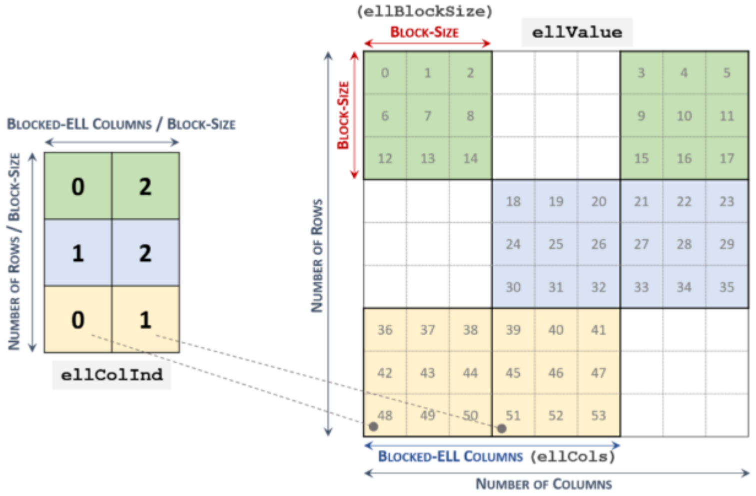
cuSPARSE Block-SpMM

- The full matrix organized in blocks and its internal memory representation
  - Compressed values and block indices
  - Not all values of matrix B are accessed for computing the output
  - Skipping unnecessary computations represented by zeros



cuSPARSE Block-SpMM

- Blocked-Ellpack Format (Blocked-ELL)
  - The right array stores nonzero values in consecutive blocks
  - The second array contains the column indices of the corresponding nonzero blocks
  - All rows in the left arrays must have the same number of blocks
  - Allow zero padding



Compressed Sparse Row (CSR) Format

- A matrix M (m * n) is represented by three 1-D vectors
- **The A vector**
  - Store values of non-zero elements
  - **Row-by-row** traversing order
- **The IA vector**
  - Store the cumulative number of non-zero elements with size m + 1
  - IA[0] = 0
  - IA[i] = IA[i - 1] + # of non-zero elements in (i-1) th row of the M
- **The JA vector**
  - Store the column index of each element in the A vector

CSR Case Study

- **A vector is [3, 4, 2, 1]**
- JA vector stores column indices of element in A
- **JA = [0, 1, 2, 1]**
- IA[0] = 0, IA[1] = IA[0] + # of non-zero elements in row 0 = 0
- IA[2] = IA[1] + 2 = 2, IA[3] = IA[2] + 1 = 3, IA[4] = IA[3] + 1 = 4
- **IA = [0, 0, 2, 3, 4]**

Index

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

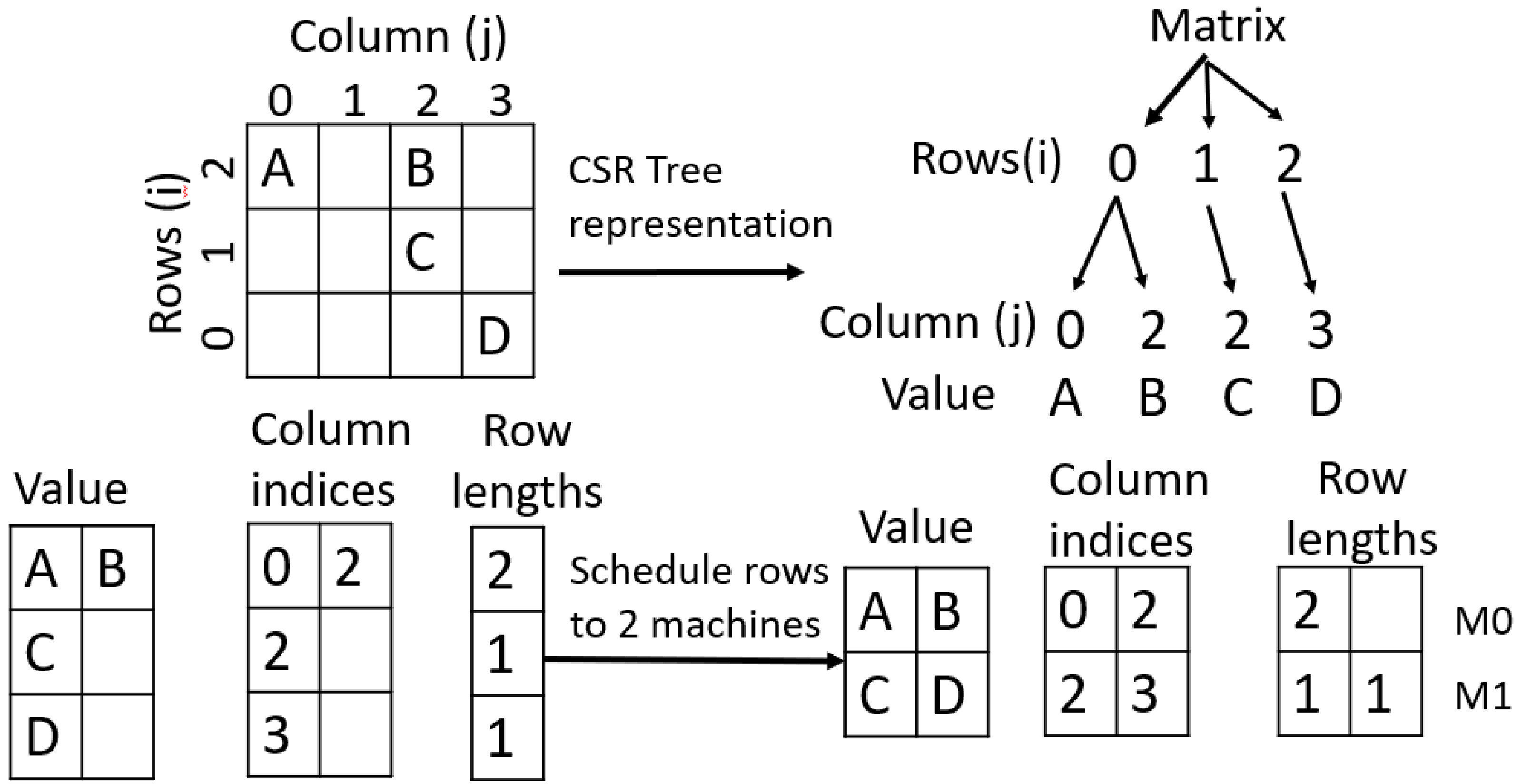
Analysis of CSR Format

- **The sparsity of the matrix**
  - (Total # of elements - # of non-zero elements) / Total # of element
- The direct array based representation required memory
  - **3 * NNZ (Number of Non-Zero)**
- CSR format required memory: **2 * NNZ + m + 1**
- CSR matrices are memory efficient as long as
  - **NNZ < (m * (n - 1) – 1) / 2**

Compressed Sparse Column (CSC) Format

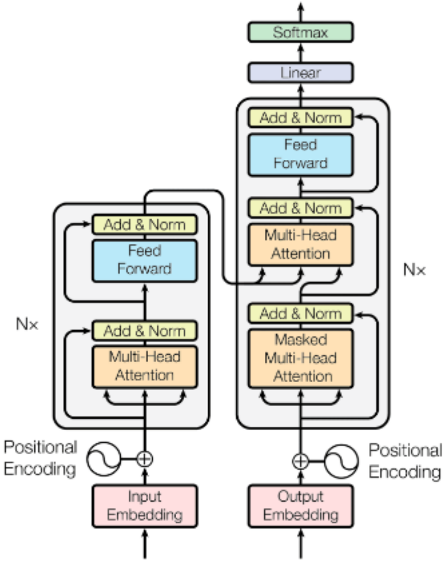
- A matrix M (m * n) is represented by three 1-D vectors
- **The A vector**
  - Store values of non-zero elements
  - **Column-by-column** traversing order
- **The IA vector**
  - Store the cumulative number of non-zero elements with size n + 1
  - IA[0] = 0
  - IA[i] = IA[I - 1] + # of non-zero elements in (i-1) th column of the M
- **The JA vector**
  - Store the row index of each element in the A vector

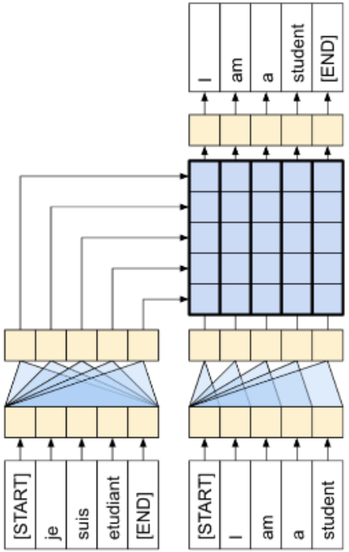
Sparse Matrix Vector Multiplication (SpMV)



TPU v4 Sparse Core

- Sparse Core accelerates Embedding layer of NLP
  - DLRM (deep learning recommendation models)



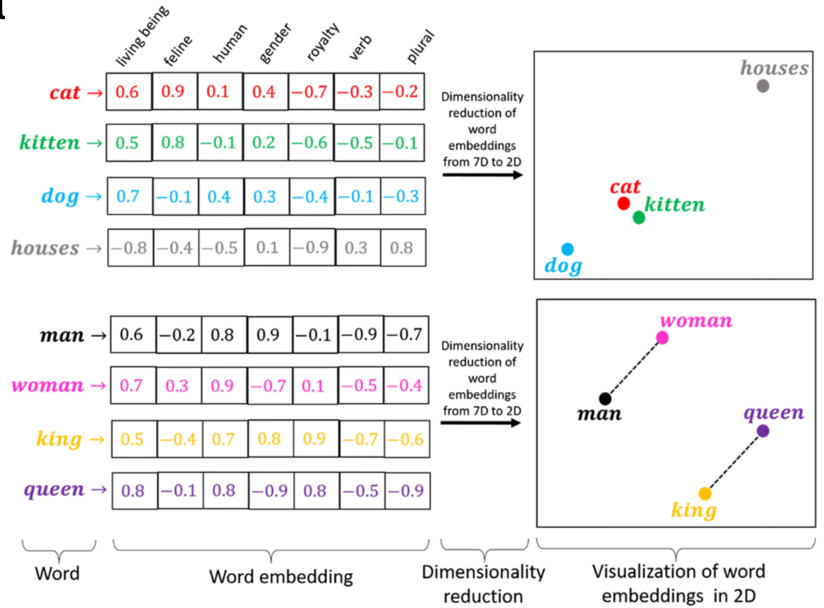


TPU v4 Sparse Core

- Accelerate embedding lookups
  - Stress memory bandwidth with all-to-all communication traffic
  - Sparse feature vector data

	1	2	3	4	5	6	7	8	9
man	1	0	0	0	0	0	0	0	0
woman	0	1	0	0	0	0	0	0	0
boy	0	0	1	0	0	0	0	0	0
girl	0	0	0	1	0	0	0	0	0
prince	0	0	0	0	1	0	0	0	0
princess	0	0	0	0	0	1	0	0	0
queen	0	0	0	0	0	0	1	0	0
king	0	0	0	0	0	0	0	1	0
monarch	0	0	0	0	0	0	0	0	1

Each word gets a 1x9 vector representation

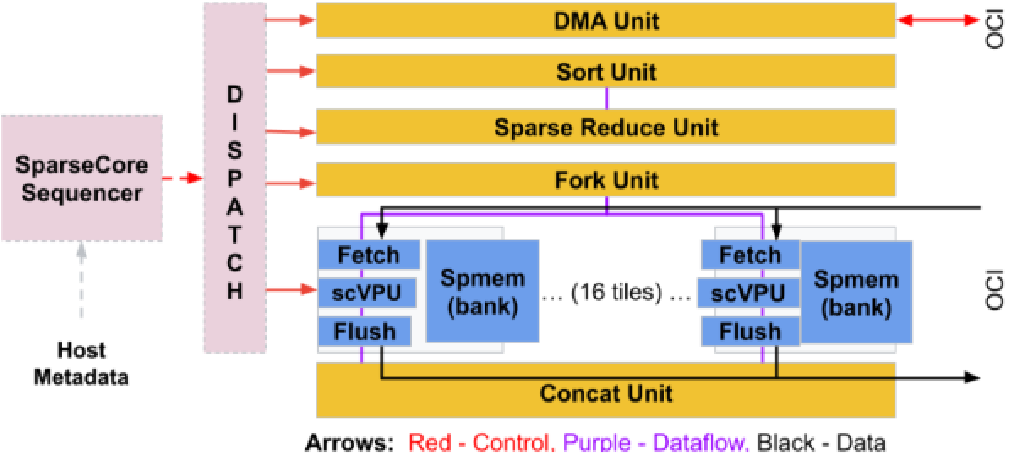


TPU v4 Sparse Core

- Problems of TensorCore for sparse embeddings
  - Produce a lot of small gather/scatter memory accesses
  - Variable length data exchange
- Dedicated inter-core connection link (ICI) with HBM memory on TPU
  - Provide fast gather/scatter memory access support

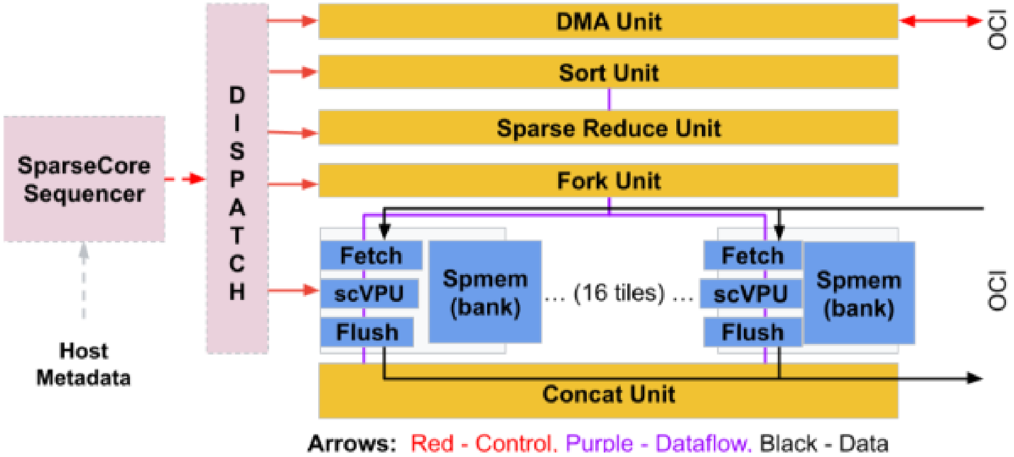
TPU v4 Sparse Core

- TPU v4 includes 4 sparse core(SC)
  - Each SC consists of 16 compute tiles
  - Each tile has an associated HBM channel
  - Support multiple outstanding memory accesses



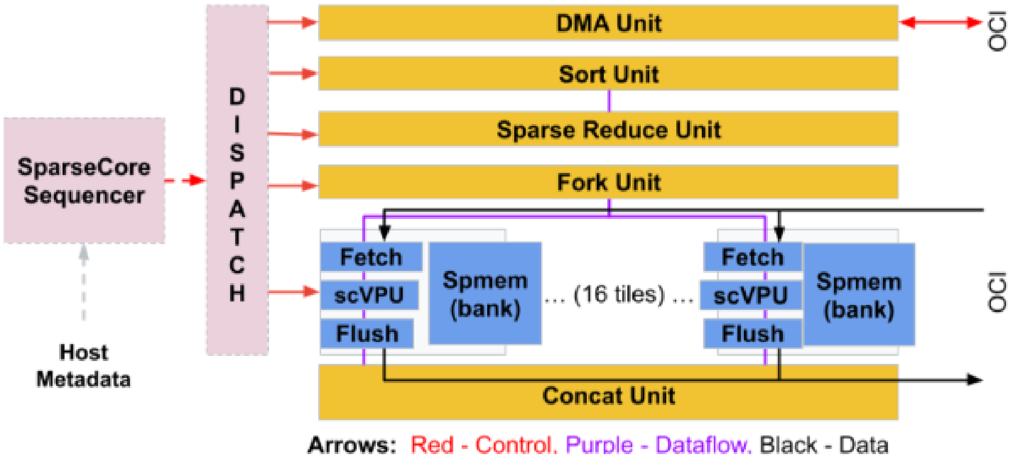
TPU v4 Sparse Core

- TPU v4 includes 4 sparse core(SC)
  - Each tile has a <u>Fetch Unit</u>, a programmable 8-wide SIMD Vector Processing Unit (<u>scVPU</u>)
  - Fetch unit reads activations and parameters from the HBM into the tile's slice of a 2.5 MB <u>spmem</u>



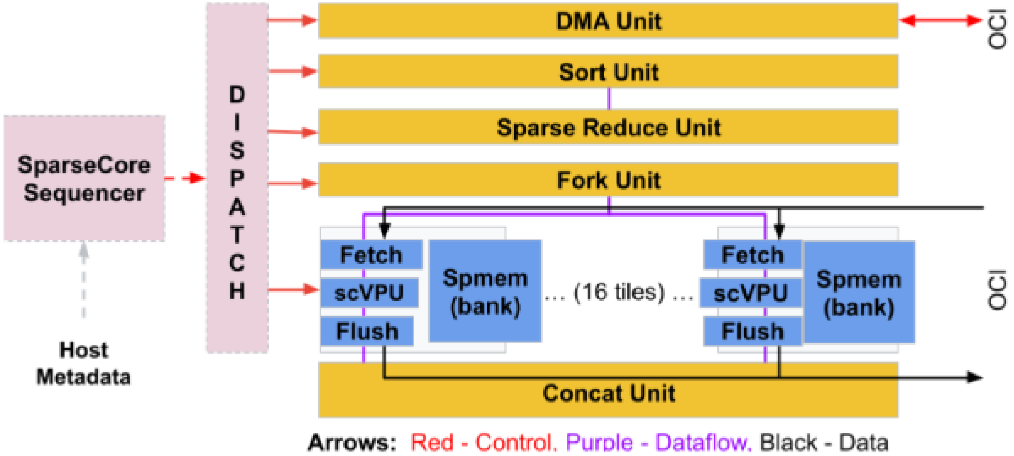
TPU v4 Sparse Core

- TPU v4 includes 4 sparse core(SC)
  - The Flush Unit writes updated parameters to HBM during the backward pass
  - Five cross-channel units (gold boxes) perform specific embedding operations

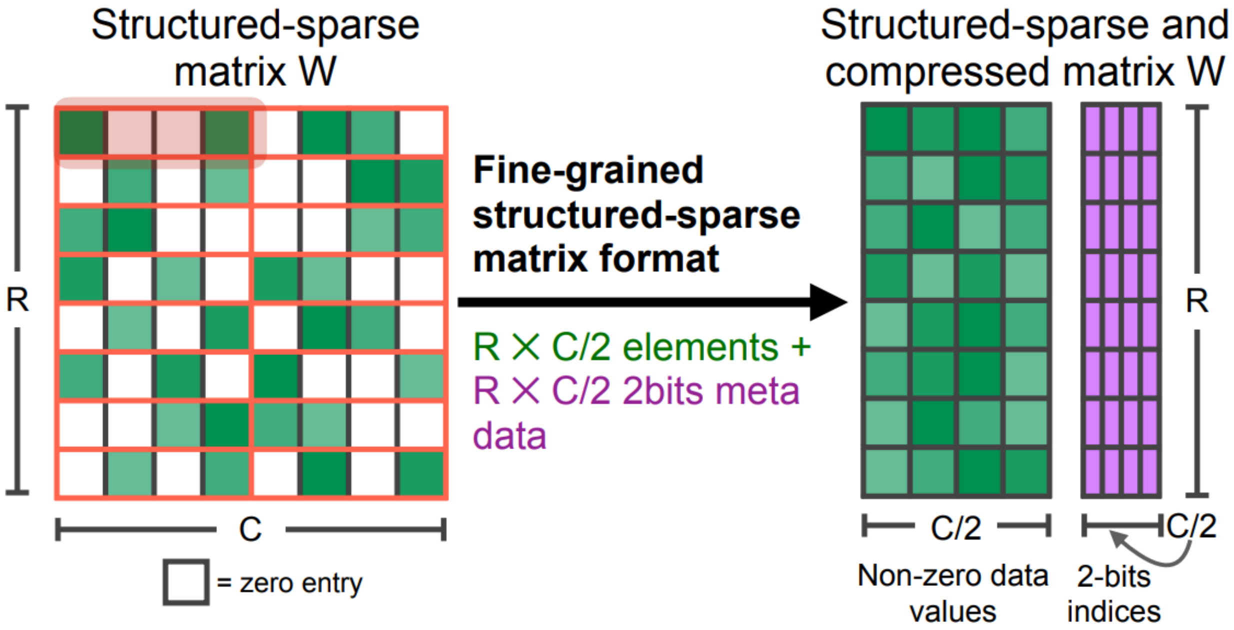


TPU v4 Sparse Core

- TPU v4 includes 4 sparse core(SC)
  - Each cross-channel unit executes CISC-like instructions
  - Operate on variable-length inputs
  - Operate across all 16 banks of spmem collectively



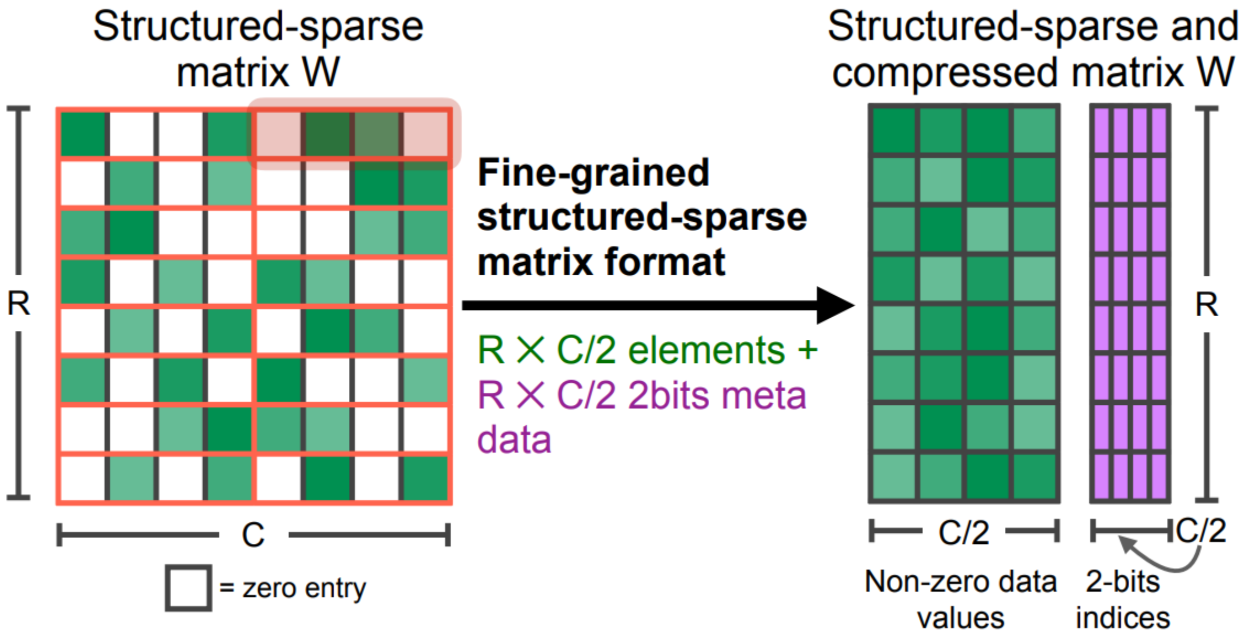
Nvidia Tensor Core: M:N Sparsity



Two weights are nonzero out of four consecutive weights (2:4 sparsity).

Accelerating Sparse Deep Neural Networks [Mishra *et al.*, arXiv 2021]

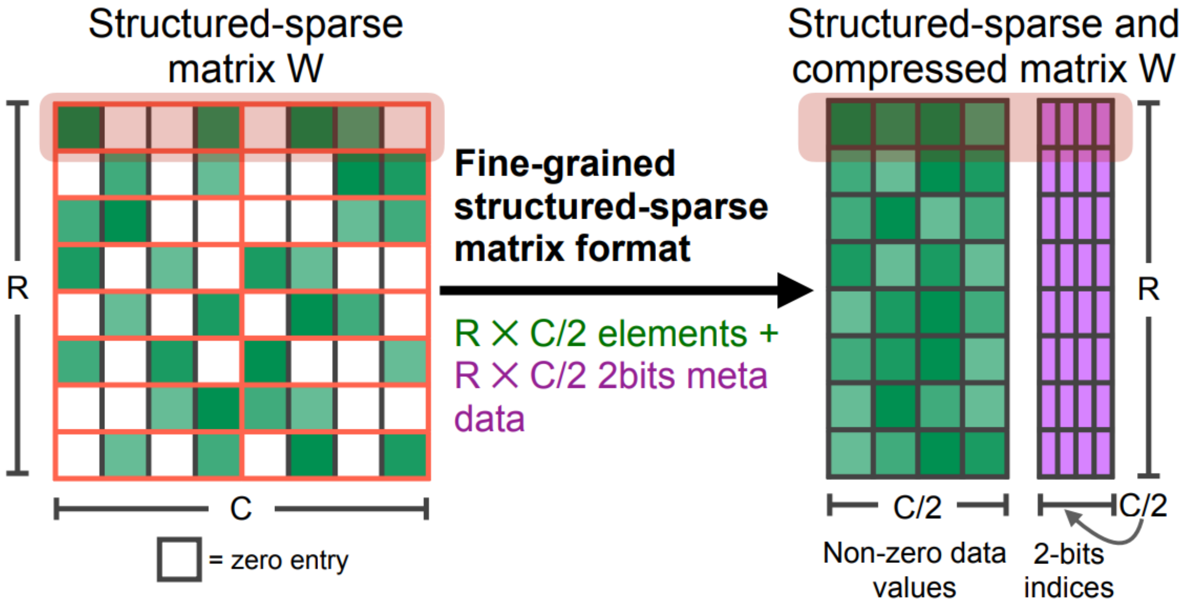
Nvidia Tensor Core: M:N Sparsity



Two weights are nonzero out of four consecutive weights (2:4 sparsity).

Accelerating Sparse Deep Neural Networks [Mishra *et al.*, arXiv 2021]

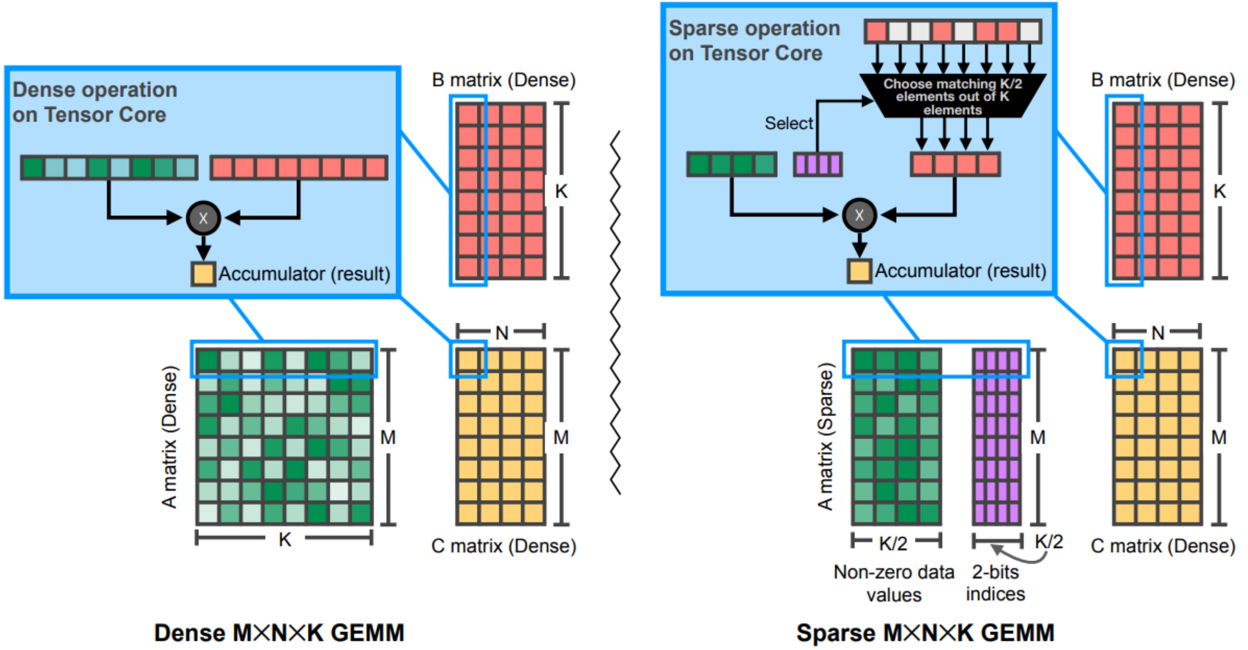
Nvidia Tensor Core: M:N Sparsity



Push all the nonzero elements to the left in memory: save storage and computation.

Accelerating Sparse Deep Neural Networks [Mishra *et al.*, arXiv 2021]

Nvidia Tensor Core: M:N Sparsity



The indices are used to mask out the inputs. Only 2 multiplications will be done out of four.

Accelerating Sparse Deep Neural Networks [Mishra *et al.*, arXiv 2021]

Nvidia Tensor Core: M:N Sparsity

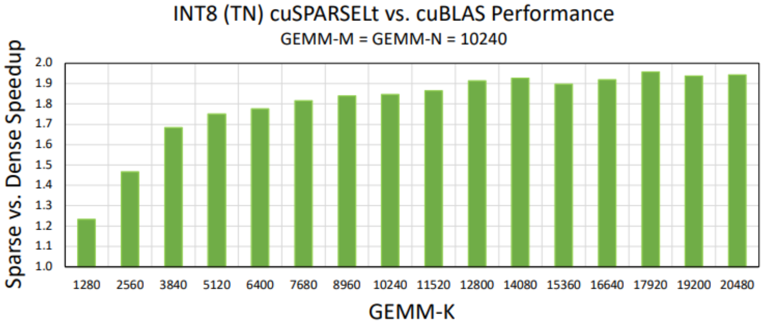


Fig. 3. Comparison of sparse and dense INT8 GEMMs on NVIDIA A100 Tensor Cores. Larger GEMMs achieve nearly a 2× speedup with Sparse Tensor Cores.

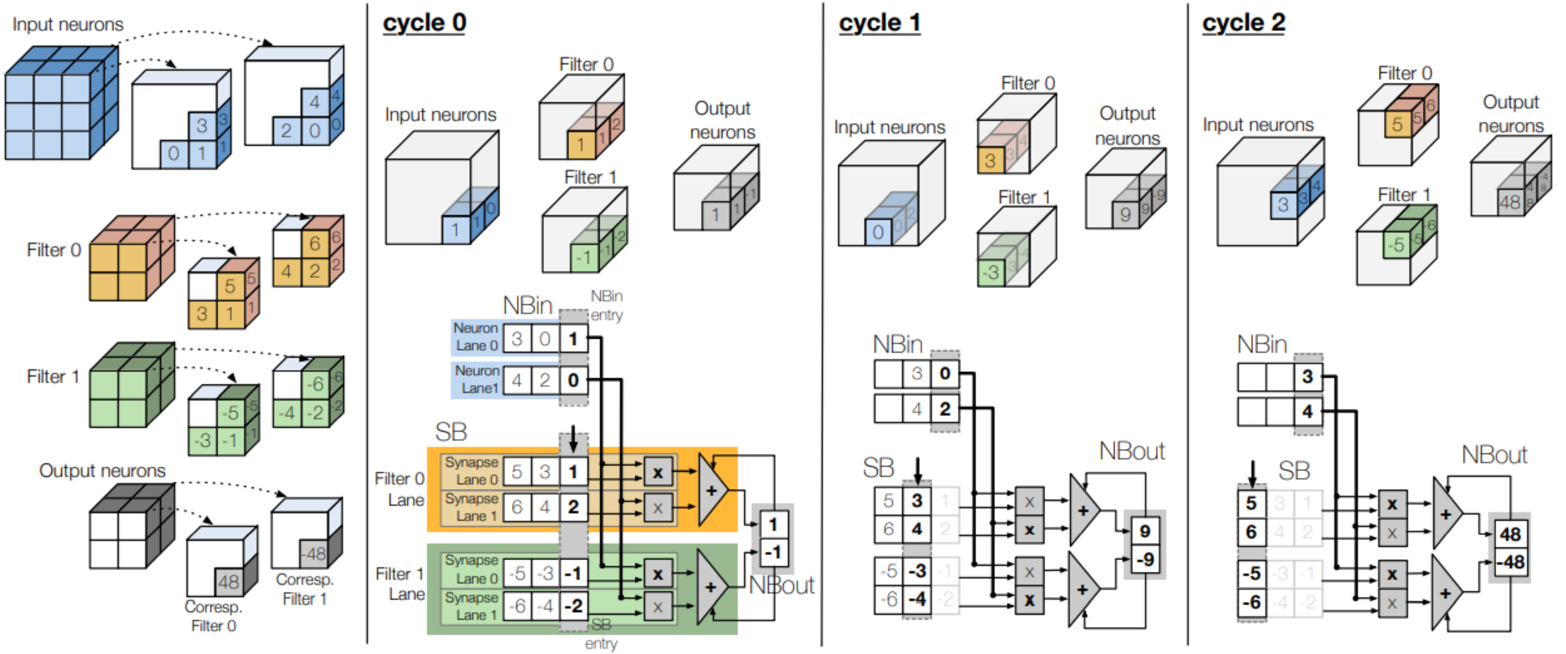
Network	Accuracy		
	Dense FP16	**Sparse FP16**	**Sparse INT8**
ResNet-34	73.7	73.9	73.7
ResNet-50	76.1	76.2	76.2
ResNet-50 (SWSL)	81.1	80.9	80.9
ResNet-101	77.7	78.0	77.9
ResNeXt-50-32x4	77.6	77.7	77.7
ResNeXt-101-32x16	79.7	79.9	79.9
ResNeXt-101-32x16 (WSL)	84.2	84.0	84.2
DenseNet-121	75.5	75.3	75.3
DenseNet-161	78.8	78.8	78.9
Wide ResNet-50	78.5	78.6	78.5
Wide ResNet-101	78.9	79.2	79.1
Inception v3	77.1	77.1	77.1
Xception	79.2	79.2	79.2
VGG-11	70.9	70.9	70.8
VGG-16	74.0	74.1	74.1
VGG-19	75.0	75.0	75.0
SUNet-128	75.6	76.0	75.4
SUNet-7-128	76.4	76.5	76.3
DRN26	75.2	75.3	75.3
DRN-105	79.4	79.5	79.4

Pruning CNNs with 2:4 sparsity will bring about large speedup for GEMM workloads and it will not incur performance drop for DNN models.

Accelerating Sparse Deep Neural Networks [Mishra *et al.*, arXiv 2021]

Cnvlutin

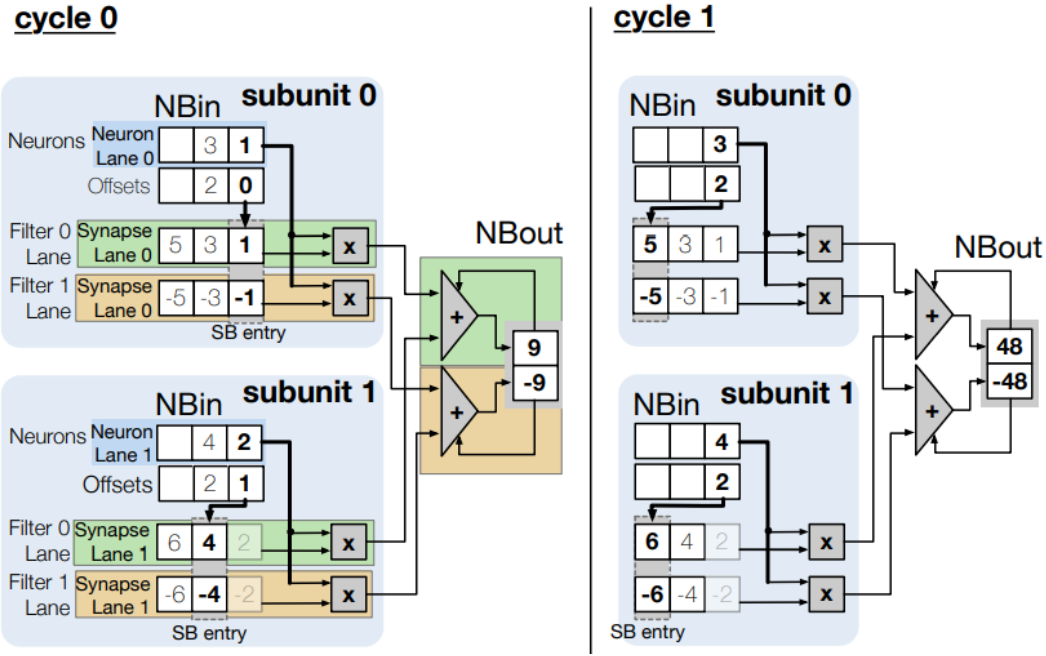
- Baseline does not skip zero and takes three cycles to complete



[Albericio et al., ISCA 2016]

Cnvlutin

- Work on CONV layer
- Cnvlutin skips zero to shorten the execution time
- Add **offset bit** to indicate the proper filter to read



[Albericio et al., ISCA 2016]

Takeaway Questions

- What are critical issues when designing a sparse DNN accelerator?
  - (A) Compressed data overhead
  - (B) Sparse data mapping
  - (C) Hard to prefetch data