Enabling Sparse Model Inference: A Micro-Kernel Aware Approach for Optimized Libraries on the RISC-V

Heng-Kuan Lee, Ph.D.
Deputy Director, RD-CA
Andes Technology

2025/07/18

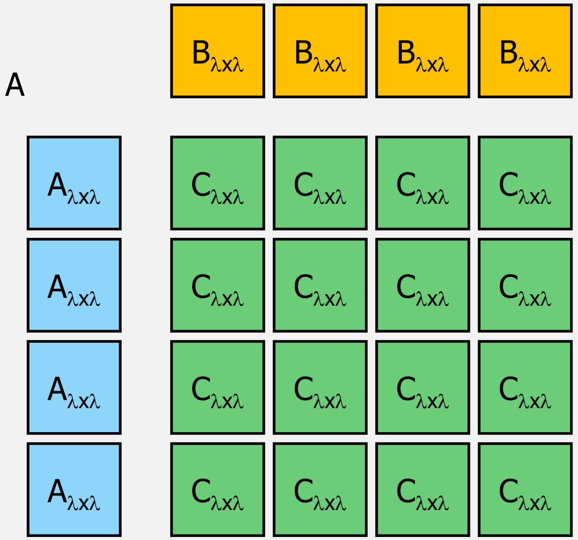
Taking RISC-V® Mainstream

ANDES™

RISC-V®

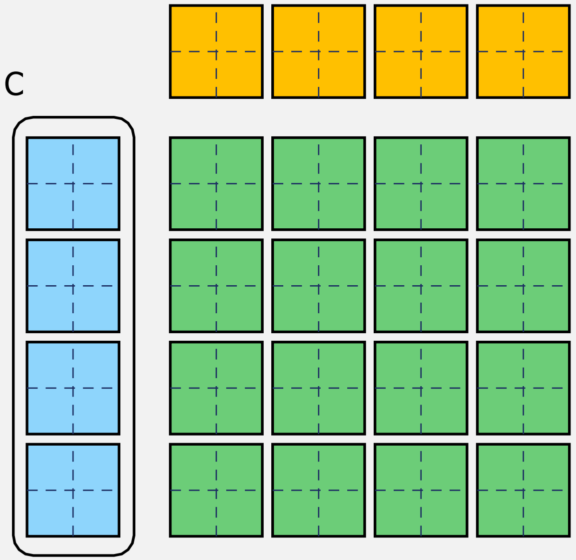
RISC-V IME and Industrial Approaches (1/3)

IME TG Option A

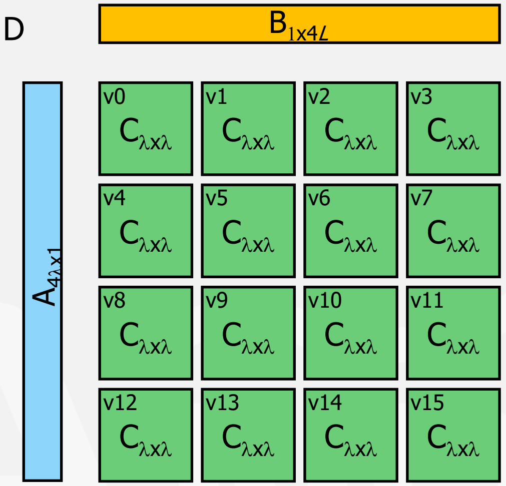


IME TG Option B

IME TG Option C

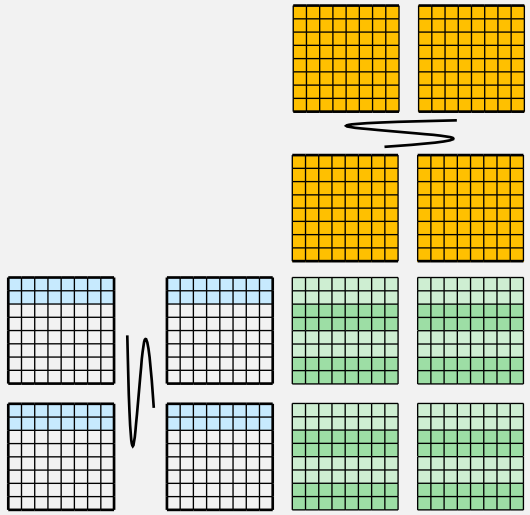


IME TG Option D

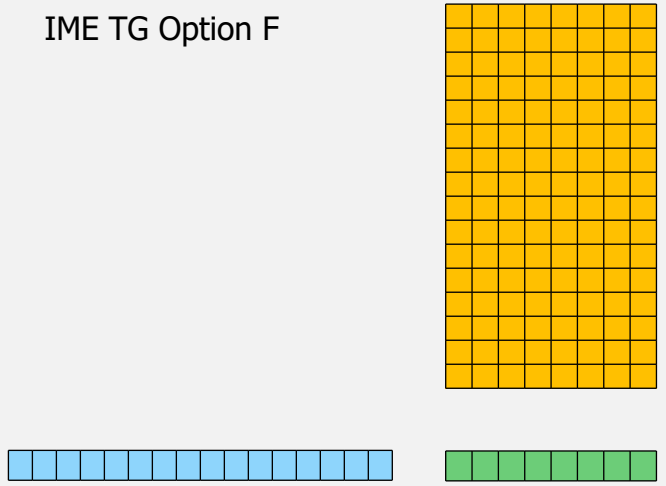


RISC-V IME and Industrial Approaches (2/3)

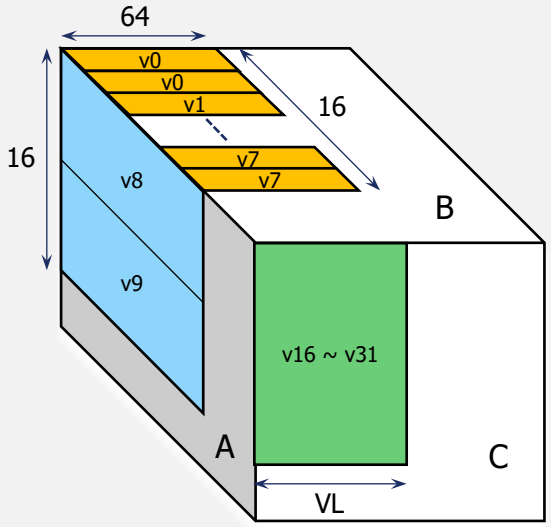
IME TG Option E



IME TG Option F

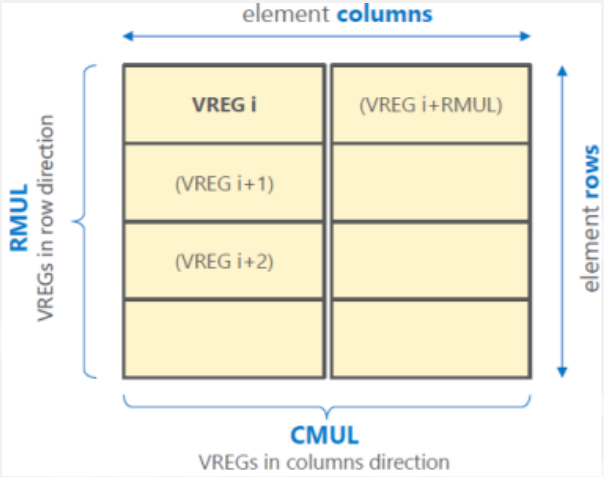


Semidynamics



IME TG Option G

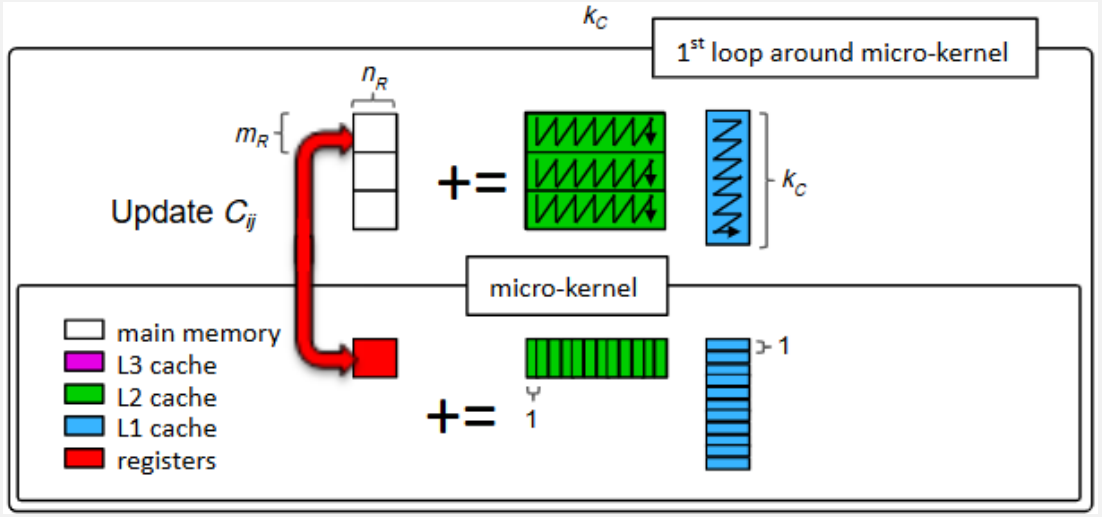
```
// example
struct TileShape {
  unsigned int rows  : MM;
  unsigned int cols  : NN;
  unsigned int rmul  :  3;
  unsigned int cmul  :  3;
  unsigned int sew   :  3;
  unsigned int mtype :  2;
  unsigned int cmo   :  1;
};
```



Taking RISC-V® Mainstream

RISC-V IME and Industrial Approaches (3/3)

- Streaming or stationary
- Column, row major
- Vector register #
- Implementation cost
  - read/write port, non-architectural states
- BLAS-like micro-kernel



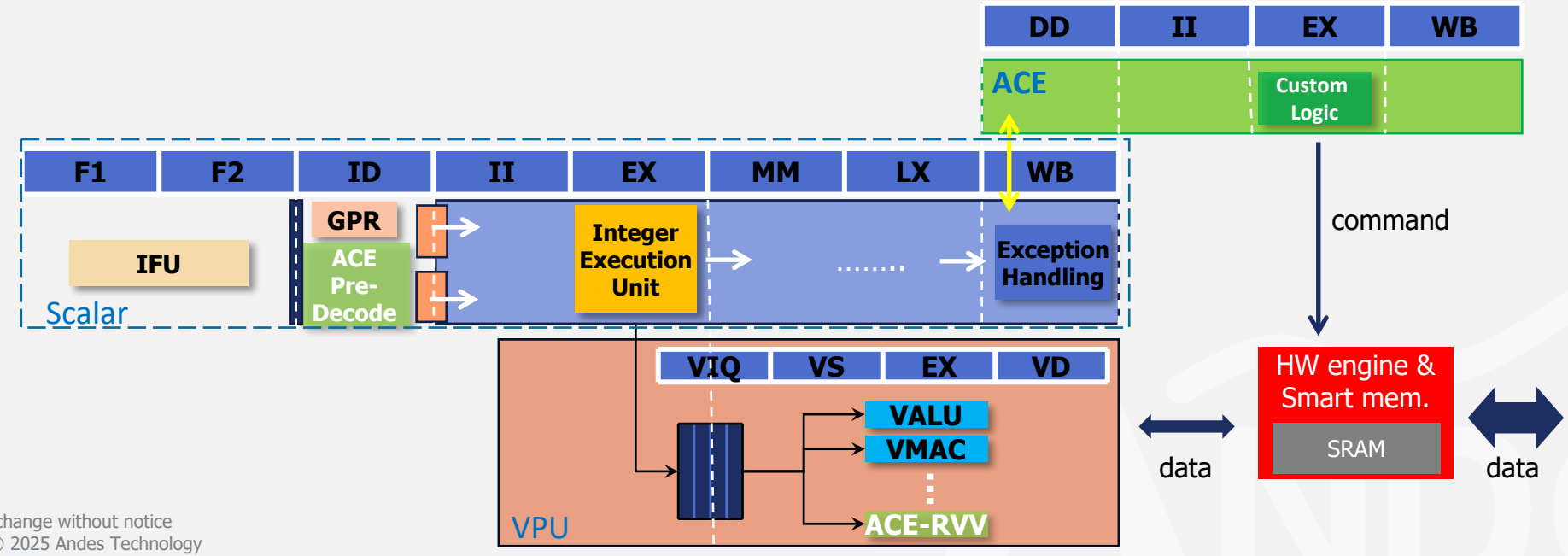
GotoBLAS algorithm in BLIS

Loop 5 for $j_c = 0 : n-1$ steps of n_c
  $\mathcal{J}_c = j_c : j_c + n_c - 1$
Loop 4 for $p_c = 0 : k-1$ steps of k_c
  $\mathcal{P}_c = p_c : p_c + k_c - 1$
  $B(\mathcal{P}_c, \mathcal{J}_c) \rightarrow \widetilde{B}_p$
Loop 3 for $i_c = 0 : m-1$ steps of m_c
  $\mathcal{I}_c = i_c : i_c + m_c - 1$
  $A(\mathcal{I}_c, \mathcal{P}_c) \rightarrow \widetilde{A}_i$
  // macro-kernel
Loop 2 for $j_r = 0 : n_c - 1$ steps of n_r
  $\mathcal{J}_r = j_r : j_r + n_r - 1$
Loop 1 for $i_r = 0 : m_c - 1$ steps of m_r
  $\mathcal{I}_r = i_r : i_r + m_r - 1$
  //micro-kernel
Loop 0 for $p_r = 0 : p_c - 1$ steps of 1
  $C_c(\mathcal{I}_r, \mathcal{J}_r) \mathrel{+}= \alpha \widetilde{A}_i(\mathcal{I}_r, p_r)\, \widetilde{B}_p(p_r, \mathcal{J}_r)$
  endfor
  endfor
  endfor
  endfor
  endfor
endfor

Taking RISC-V® Mainstream

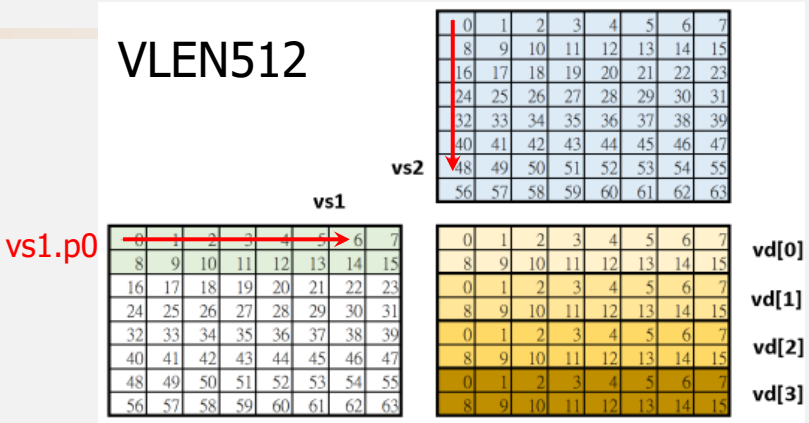
Custom Vector Instructions

- RISC-V ISA extension enables new instructions, coprocessors, and memory locations
- Custom extension
  - ACE: Andes Custom Extension
  - ACE-RVV: ACE for RVV
- Andes's ACE and ACE-RVV facilitate customer implementation of IME or AME
- Let's explore implementing IME /w sparse inference based on Option E

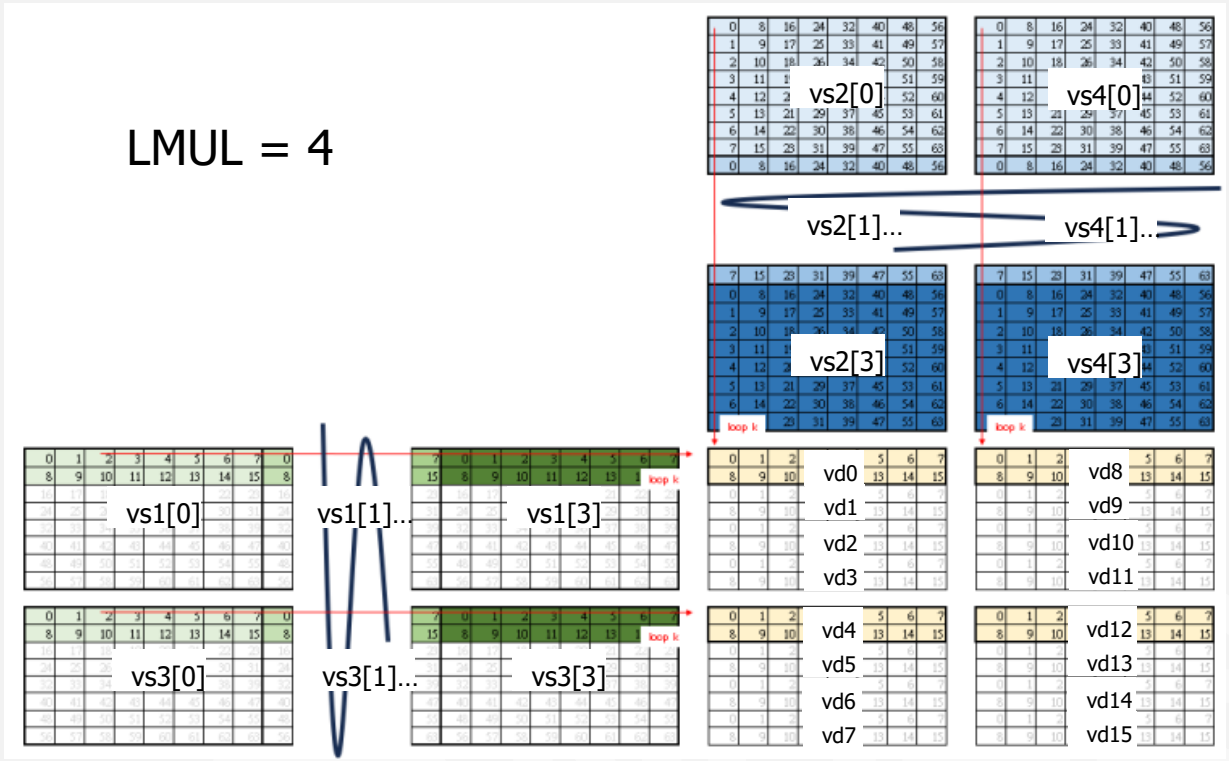


Recap RISC-V IME TG Option E

- Considering Quantized Model as int32 += int8*int8
  - amm vd, vb, va
    - vd[0] = vs1.p0 * vs2




```
vld.2d vs2,rs2,rs4,imms;
vld.2d vs4,rs6,rs8,imms;
while(k>0){
   vld.ef vs1,rs1,rs3,imms;
   amm    vd0,vs2[0],vs1[0];
   amm    vd0,vs2[1],vs1[1];
   amm    vd0,vs2[2],vs1[2];
   amm    vd0,vs2[3],vs1[3]||vld.ef vs1,rs1,rs3,imms;
   amm    vd0,vs2[0],vs1[0];
   amm    vd0,vs2[1],vs1[1];
   amm    vd0,vs2[2],vs1[2];
   amm    vd0,vs2[3],vs1[3]||vld.ef vs1,rs1,rs3,imms;
 …unroll…
   vld.2d vs2,rs2,rs4,imms;
   vld.2d vs4,rs6,rs8,imms;
   k -= Ktile*unroll_factor;
}
```

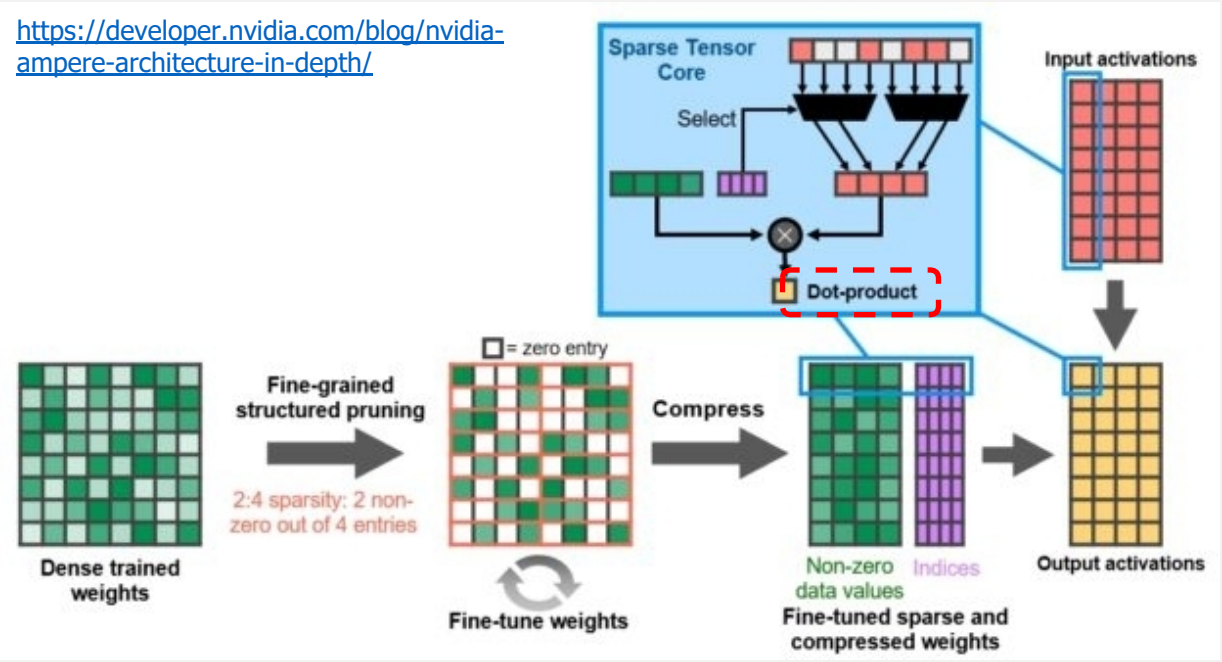


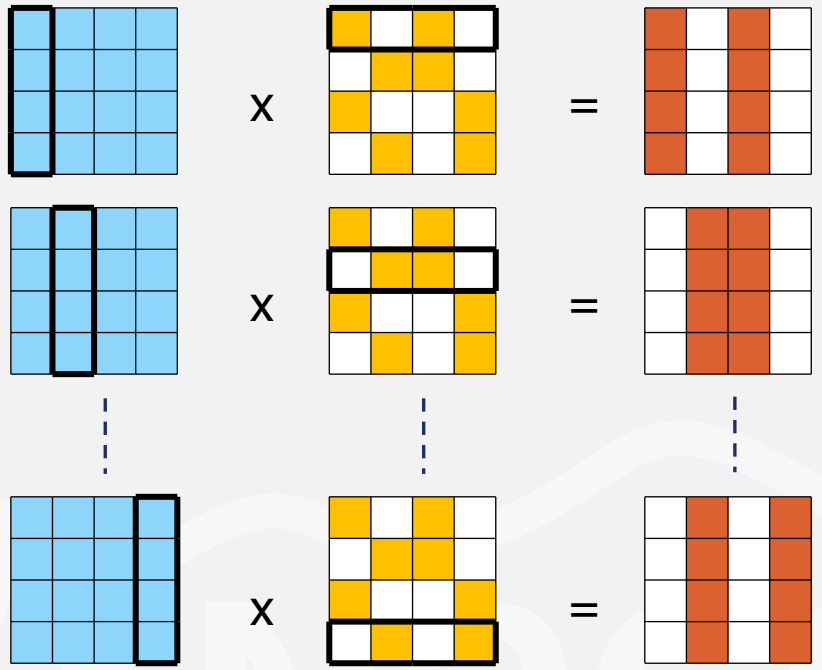
Taking RISC-V® Mainstream

Block-wise Sparsity: The Target Problem

- Well-known pixel-wise n:m sparsity (nVidia A100)
  - ☹ Fundamental incompatibility with outer product implementation
  - ☹ High hardware specialization requirement
  - ☹ Overhead of non-zero indices & reduced effective compression ratio
  - ☺ Superior fine-grained pruning for accuracy preservation

https://developer.nvidia.com/blog/nvidia-ampere-architecture-in-depth/

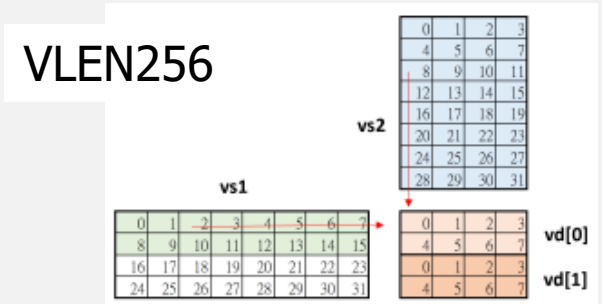


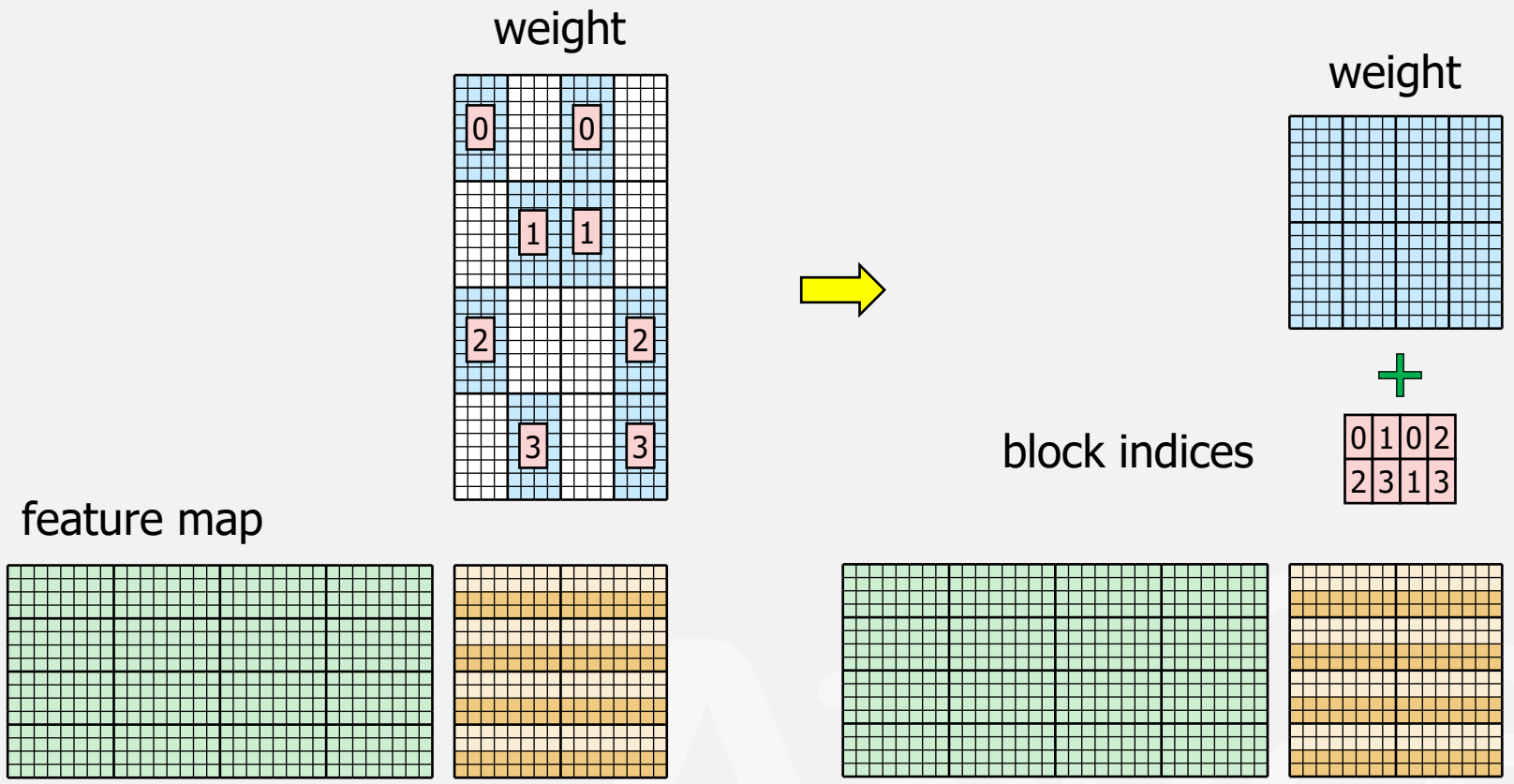


Outer product /w 2:4 sparsity

Block-wise Sparsity

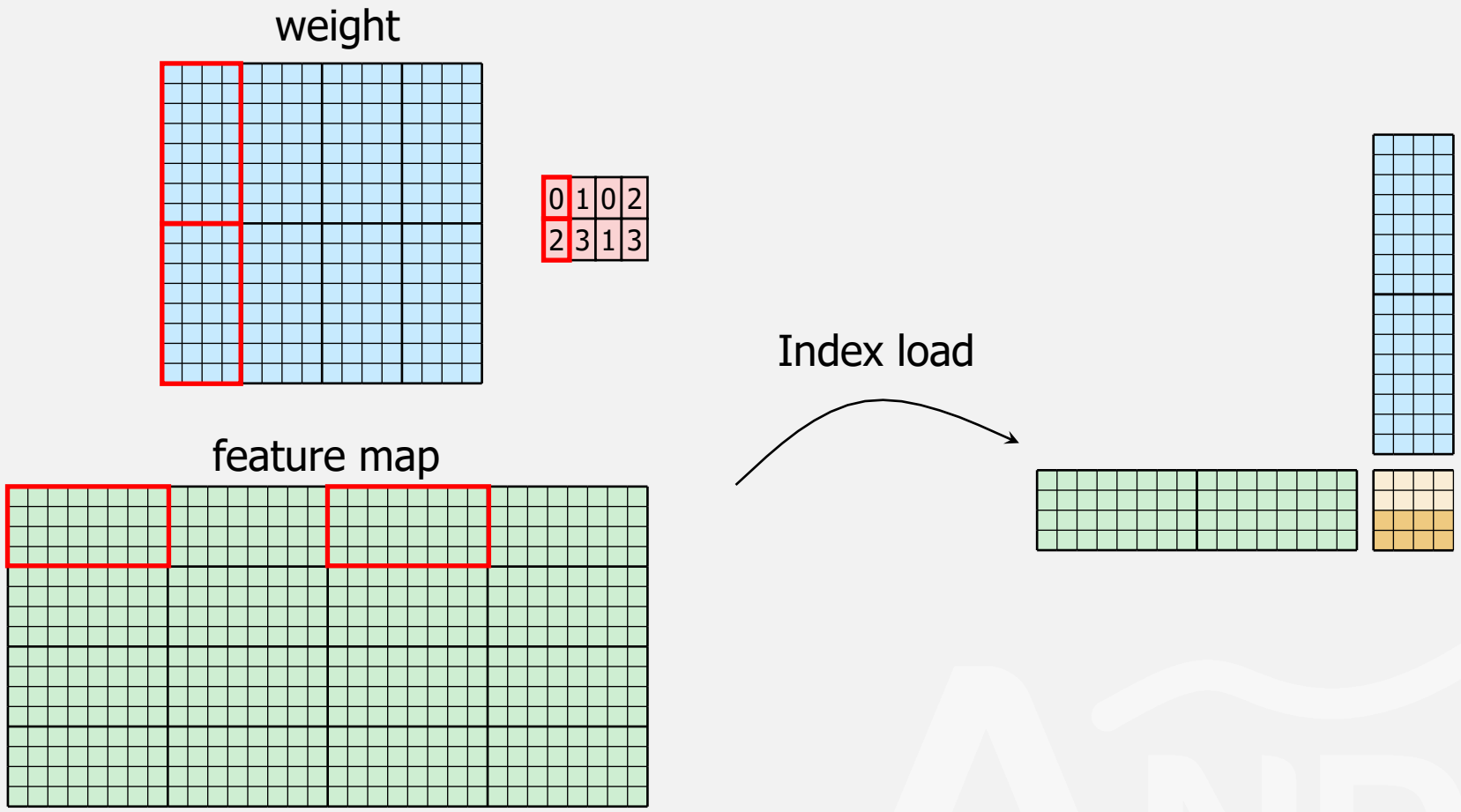
- If the block size of sparsity can align with the instruction-level tile
  - VLEN = 256, LMUL = 4, the instruction-aligned sparsity block can be shown as below
- Block 2:4 sparsity





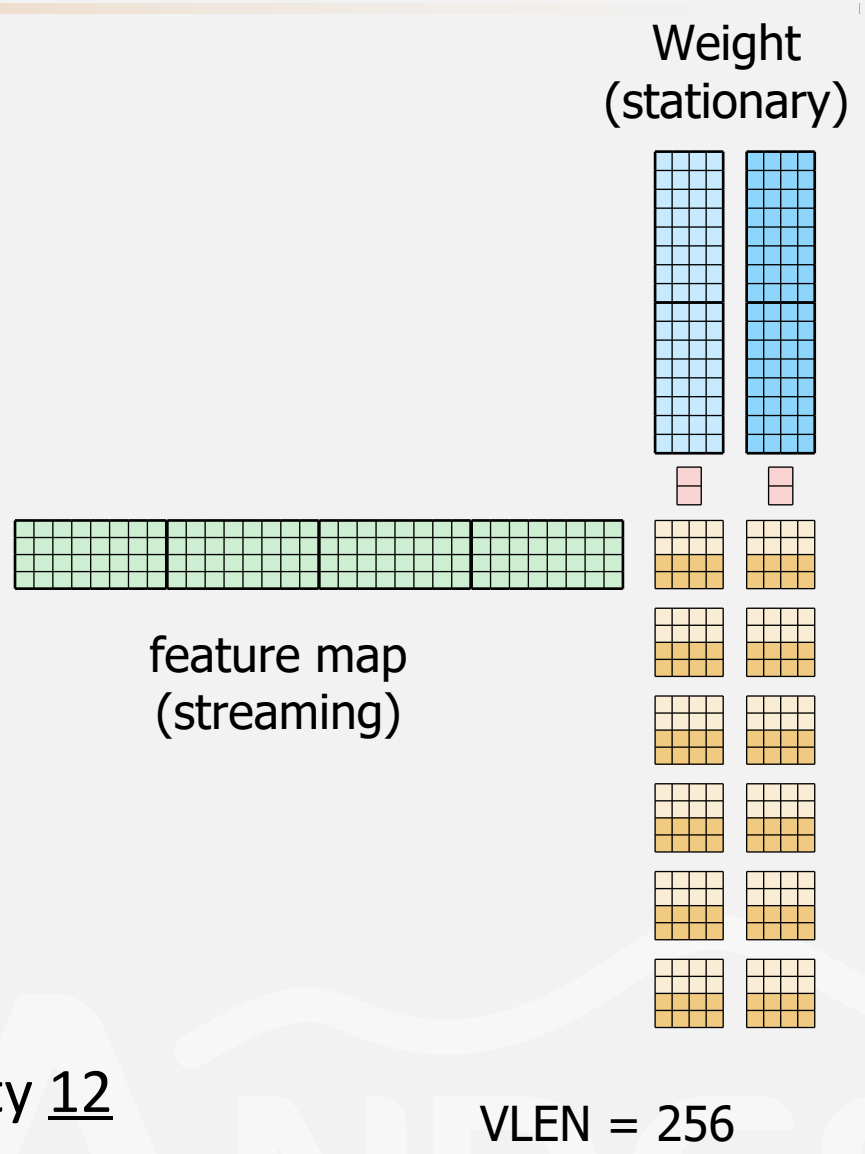
Enhanced IME: Our Core Instruction-Level Solution

- IME matrix multiplication + index load instruction
- High level flow



Micro-Kernel Aware Execution Flow

- Block-wise sparsity aligns with instruction-level tile
- Micro-kernel: Orchestrating peak performance
  - Key Optimization Strategies
    - Loop Unrolling
    - Cache line aligned for index load
    - Maximized vector register utilization
  - Compute intensity
    - VLEN/DLEN 256/256, int8
      - Baseline (Dense): 5.71
      - Dense-Equivalent /w sparsity: 7.47
        - Actual Sparse: 3.733
    - VLEN/DLEN 512/512, int8
      - Baseline 8 to Dense-Equivalent /w sparsity 12



Pruning Experiment Result (1/3)

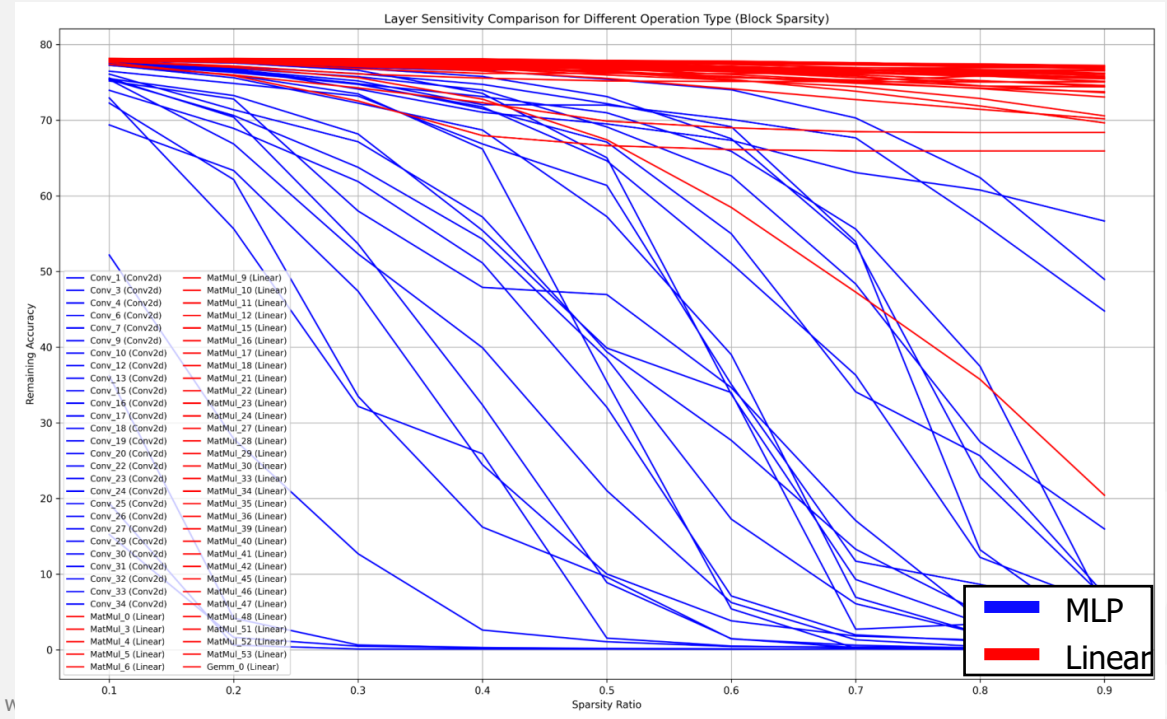
- Model: ResNet50
- Considering VLEN 256 → block 4x8 X 8x4
- Finetune epoch: 30

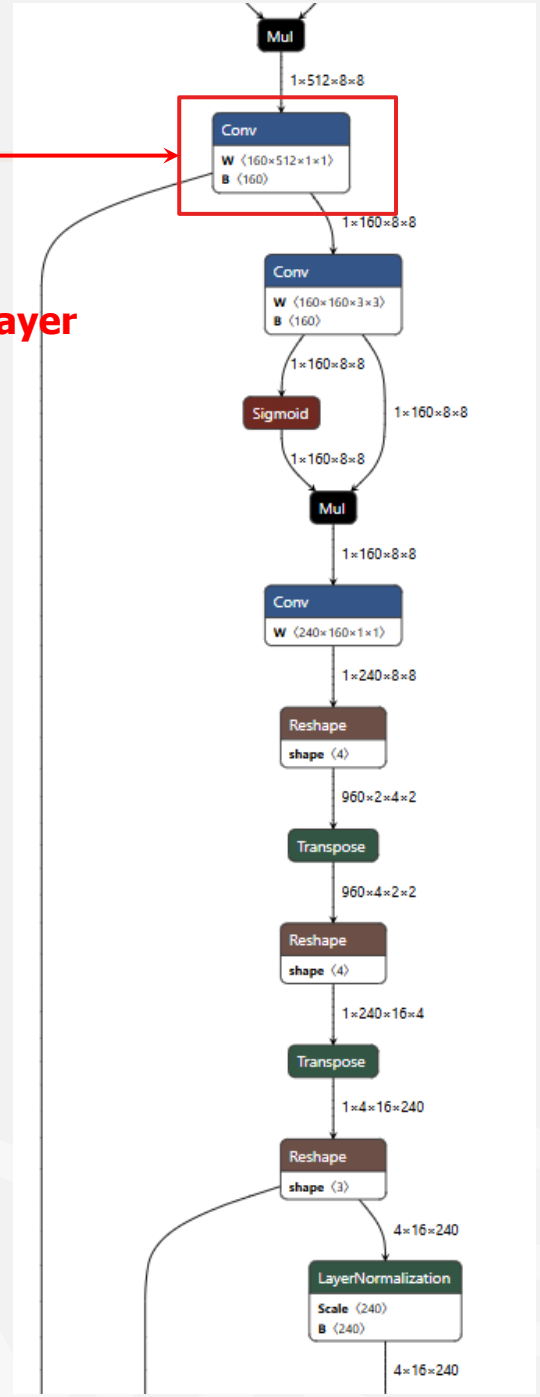
	Baseline Acc	Pruned Acc	Param Sparsity
Structured pruning	76.15%	74.95% (1.20 ↓)	45.02%
Semi-Structured (2:4)		76.25% (0.10 ↑)	45.87%
Block sparsity (4x8)	76.15%	75.57% (0.58 ↓)	45.87%
Block 2:4 sparsity (4x8)		**75.52% (0.63 ↓)**	**45.87%**

Pruning is not applied to 1st and last layers

Pruning Experiment Result (2/3)

- Model: MobileViT
- Sensitivity Analysis
- Conv is more sensitive than FF
  - MLP parameters #: 63.58%
  - Conv parameters #: 36.28%





Pruning Experiment Result (3/3)

- Model: MobileViT
- Considering VLEN 256 → block 4x8 X 8x4
- Finetune epoch: 30

		Baseline Acc	Pruned Acc	Param Sparsity
Block 2:4 sparsity (4x8)	(all layers)		71.39% (5.83 ↓)	43.91%
Block 2:4 sparsity (4x8)	**(MLP only)**	77.216%	**76.02% (1.20 ↓)**	**31.70%**
Block 2:4 sparsity (4x8)	**(MLP only, excepting last MLP)**		**76.52% (0.70 ↓)**	**26.00%**

Conv layer is much more sensitive than linear layer in VIT

Discussion and Future Directions

- LLM Challenges: Compute & Bandwidth:
  - Despite MoE, LLMs' FFNs demand high compute & memory bandwidth.
  - Sparsity (50-90% observed) directly mitigates these bottlenecks.
- Advancing Sparsity on RISC-V Matrix Extension:
  - Our Micro-Kernel Aware approach, with Block N:M Sparsity, accelerate various RISC-V matrix extension options (e.g., IME/VME/AME).
  - Next steps: More aggressive, dynamic sparsity (e.g., hierarchical sparsity, dual-side).

Llama-3.2 1B: Parameter Count

Layer	Size	Heads	Layers	Parameters
Embedding	128,256 x 2,048		1	262.7 E6
W_K (Key)	2,048 x 16	32	16	16.7 E6
W_Q (Query)	2,048 x 64	32	16	67.1 E6
W_V (Value)	2,048 x 16	32	16	16.7 E6
W_O (Output)	2,048 x 2,048		16	67.1 E6
MLP Gate Projection	**2,048 x 8,192**		**16**	**262.7 E6**
MLP Up Projection	**2,048 x 8,192**		**16**	**262.7 E6**
MLP Down Projection	**8,192 x 2,048**		**16**	**262.7 E6**

Llama-3.1 405B: Parameter Count

Layer	Size	Heads	Layers	Parameters
Embedding	128,256 x 16,384		1	2 E9
W_K (Key)	16384 x 128	128	126	2 E9
W_Q (Query)	16384 x 128	128	126	34 E9
W_V (Value)	16384 x 128	128	126	2 E9
W_O (Output)	16384 x 16384		126	34 E9
MLP Gate Projection	**16384 x 53248**		**126**	**110 E9**
MLP Up Projection	**16384 x 53248**		**126**	**110 E9**
MLP Down Projection	**53248 x 16384**		**126**	**110 E9**
Unembedding	16384 x 128256		1	2 E9