

Systematic Modeling and Design of Sparse Tensor Accelerators

Yannan (Nellie) Wu

May 5, 2023

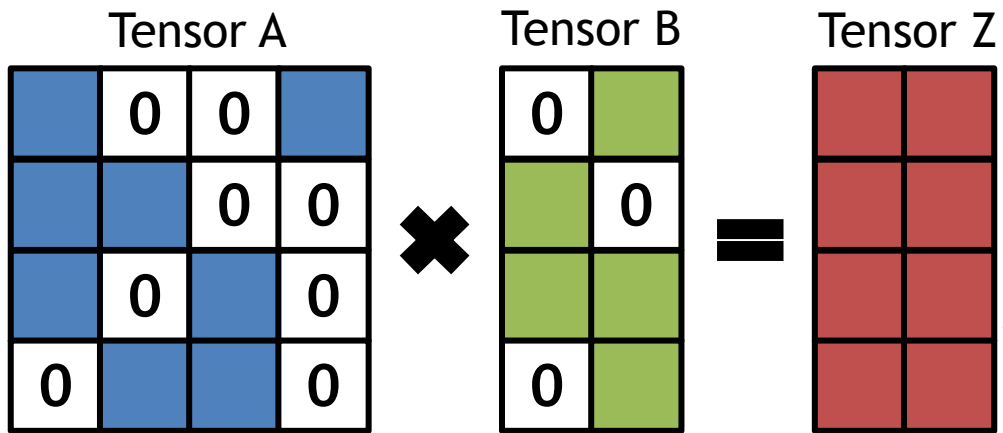
Thesis Committee:

Vivienne Sze

Joel Emer

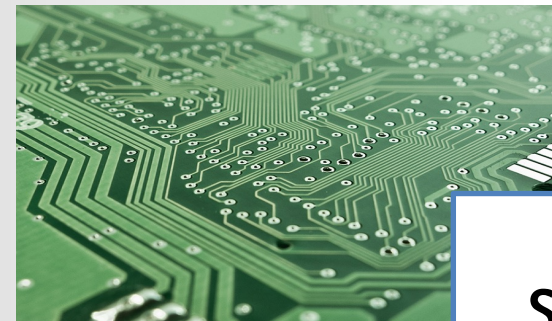
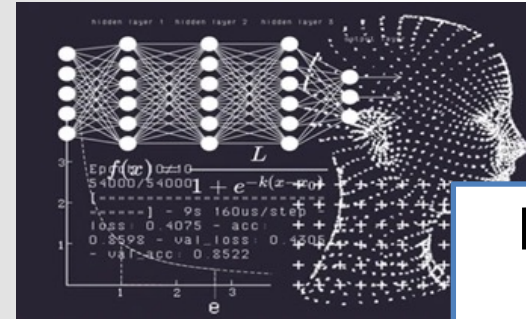
Daniel Sanchez

Sparse Tensor Algebra is Used in Many Applications



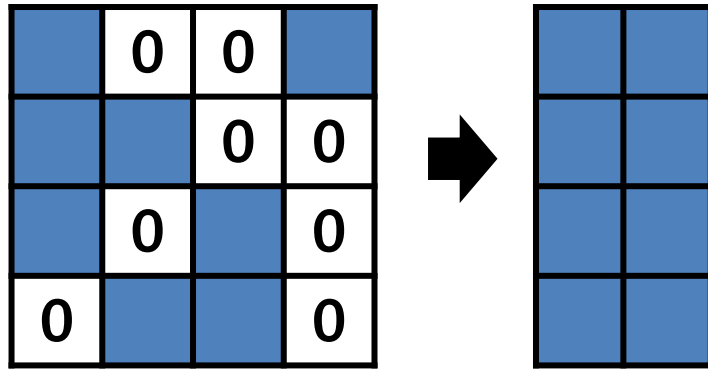
Sparse Tensor Algebra
(e.g., sparse matrix multiplication)

**Inefficient processing on
general-purpose processors**



Sparsity Introduces Opportunities for Hardware Savings

Sparse Tensor Accelerators



Zero Values
Can be Compressed Away

$$x \times 0 = 0$$

$$x + 0 = x$$

Ineffectual Computations
Can be Eliminated

Important to develop next-generation sparse tensor accelerators
to exploit such opportunities



Goal of this Thesis

- ❑ Identify the challenges for developing next-generation sparse tensor accelerators
- ❑ Propose solutions to the challenges

Challenges

Develop Efficient and Flexible Hardware for Diverse Sparsity Characteristics

Challenges

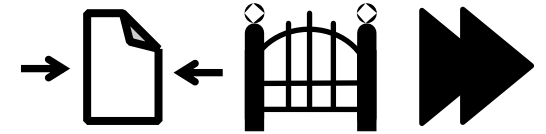
Qualitatively Learn Insights from Existing Designs

Quantitatively Compare Existing Designs and Explore New Designs

Develop Efficient and Flexible Hardware for Diverse Sparsity Characteristics

Proposed Solutions

I. Systematic Understanding of Design Space

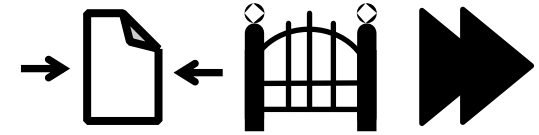


Quantitatively Compare Existing Designs and Explore New Designs

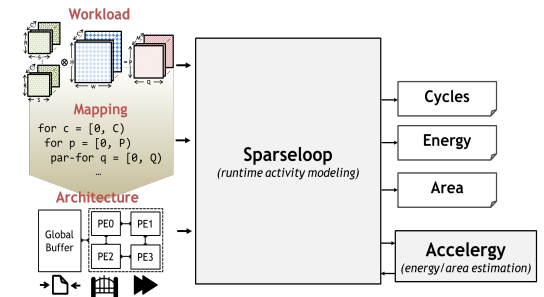
Develop Efficient and Flexible Hardware for Diverse Sparsity Characteristics

Proposed Solutions

I. Systematic Understanding of Design Space



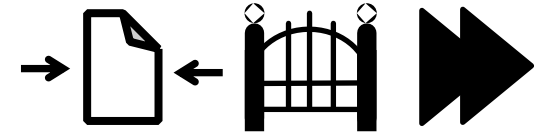
II. Sparseloop Modeling Infrastructure



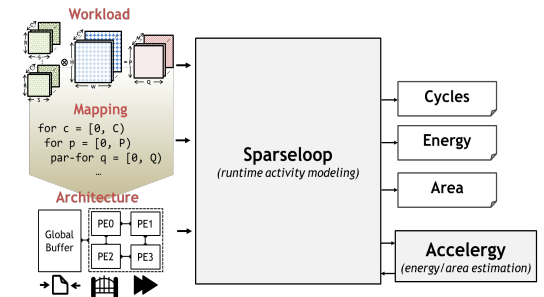
Develop Efficient and Flexible Hardware for Diverse Sparsity Characteristics

Proposed Solutions

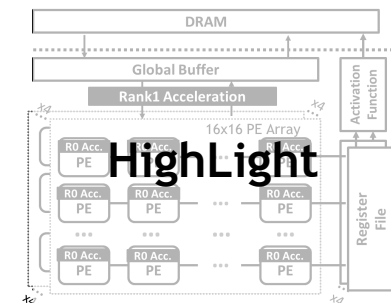
I. Systematic Understanding of Design Space



II. Sparseloop Modeling Infrastructure



III. Efficient and Flexible Accelerator Designs



Key Thesis Contributions

I. Systematic Understanding of Design Space

- Sparse Acceleration Feature (SAF) Taxonomy
- Systematic Design Description

II. Sparseloop Modeling Infrastructure

- Statistical Workload Characterization for Fast Modeling Speed
- Modularized Modeling Procedure for Tractable Complexity
- Modularized SAF Impact Analysis for Tractable Complexity
- Example Early Design Space Exploration
- Flexible Energy Estimation Backend
- Open-Source Codebase

III. Efficient and Flexible Accelerator Designs

- Hierarchical Structured Sparsity (HSS) in DNNs
- Precise Sparsity Pattern Specification
- Modularized Acceleration Design Methodology for HSS
- Compression for HSS
- Hierarchical skipping with Variable Length Fetch Support
- Speculative Tiling by Overbooking Buffer Capacity (*joint work with Fisher Xue*)

What We Will Cover Today

I. Systematic Understanding of Design Space

- Sparse Acceleration Feature (SAF) Taxonomy
- Systematic Design Description

II. Sparseloop Modeling Infrastructure

- Statistical Workload Characterization for Fast Modeling Speed
- Modularized Modeling Procedure for Tractable Complexity
- Modularized SAF Impact Analysis for Tractable Complexity
- Example Early Design Space Exploration
- Flexible Energy Estimation Backend
- Open-Source Codebase

III. Efficient and Flexible Accelerator Designs

- Hierarchical Structured Sparsity (HSS) in DNNs
- Precise Sparsity Pattern Specification
- Modularized Acceleration Design Methodology for HSS
- Compression for HSS
- Hierarchical skipping with Variable Length Fetch Support
- Speculative Tiling by Overbooking Buffer Capacity (*joint work with Fisher Xue*)

Summary of Publications

- **Modeling Methodology**

- Y. N. Wu, J. S. Emer, V. Sze, *Accelergy: An Architecture-Level Energy Estimation Methodology for Accelerator Designs*, ICCAD, 2019
- Y. N. Wu, V. Sze, J. S. Emer, *An Architecture-Level Energy and Area Estimator for Processing-In-Memory Accelerator Designs*, ISPASS, 2020
- F. Wang, Y. N. Wu, M. Woicik, V. Sze, J. S. Emer, *Architecture-Level Energy Estimation for Heterogeneous Computing Systems*, ISPASS, 2020
- Y. N. Wu, P.-A. Tsai, A. Parashar, V. Sze, J. S. Emer, *Sparseloop: An Analytical, Energy-Focused Design Space Exploration Methodology for Sparse Tensor Accelerators*, ISPASS, 2021
- Y. N. Wu, P.-A. Tsai, A. Parashar, V. Sze, J. S. Emer, *Sparseloop: An Analytical Approach to Sparse Tensor Accelerator Modeling*, MICRO, 2022
- M. Gilbert, Y. N. Wu, A. Parashar, V. Sze, J. S. Emer, *LoopTree: Enabling Exploration of Fused-layer Dataflow Accelerators*, ISPASS, 2023

- **Accelerator Design**

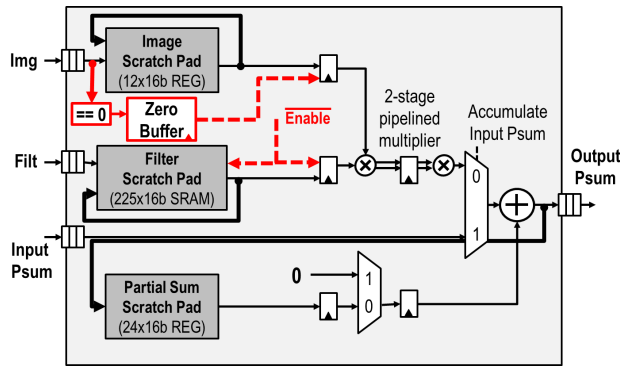
- Y. N. Wu, S. Muralidharan, P.-A. Tsai, A. Parashar, V. Sze, J. S. Emer, *HighLight: Efficient and Flexible DNN Acceleration with Hierarchical Structured Sparsity*, MICRO, 2023, under submission
- Z. Y. Xue, Y. N. Wu, J. S. Emer, V. Sze, *Accelerating Sparse Tensor Algebra by Overbooking Buffer Capacity*, MICRO, 2023, under submission

I. Systematic Understanding of Sparse Tensor Accelerator Design Space

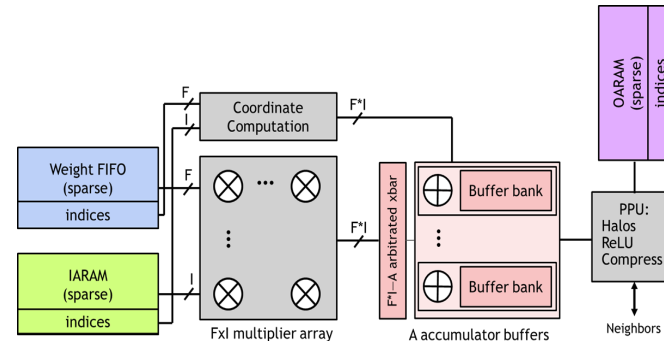
- Sparse Acceleration Feature Taxonomy
- Systematic Design Description

There Exist Ample Sparse Tensor Accelerators

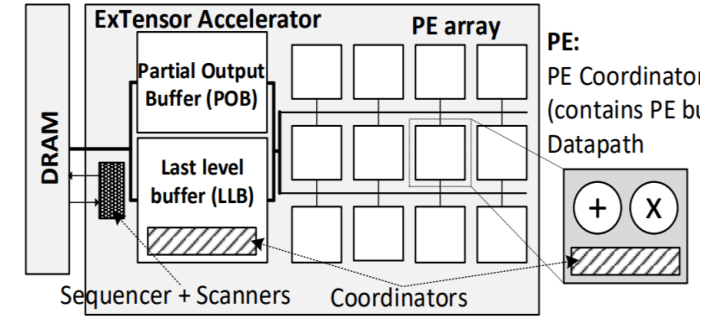
Sampled accelerator designs



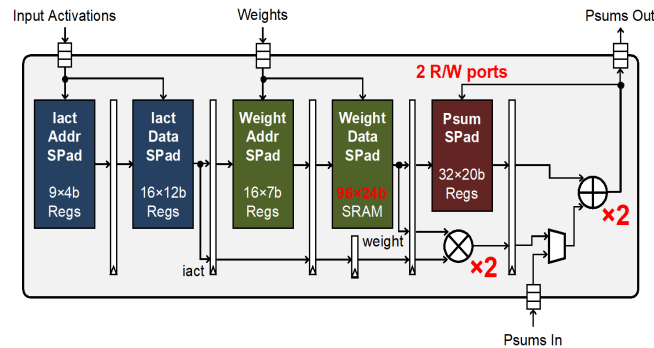
Eyeriss [JSSC2017]



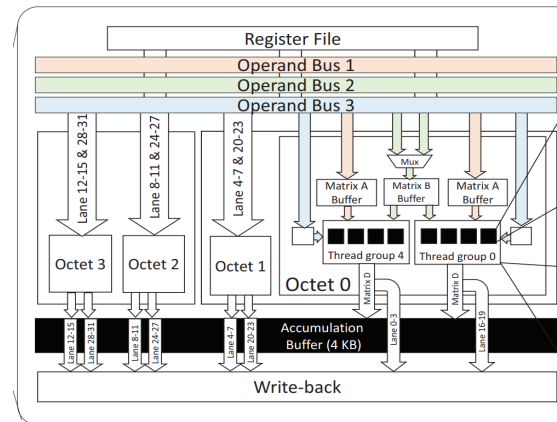
SCNN [ISCA2017]



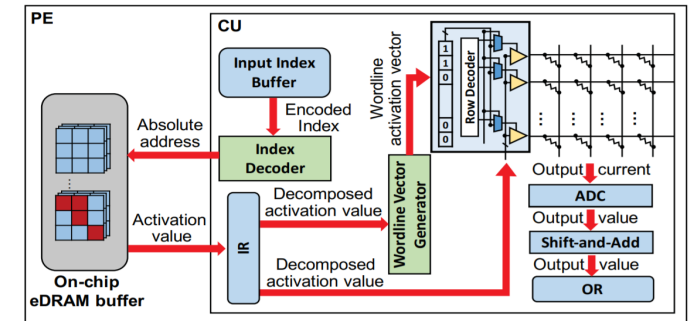
ExTensor [MICRO2019]



Eyeriss V2 [JETCAS2019]



DSTC [ISCA2021]



Sparse-ReRAM [ISCA2019]

There Exist Ample Sparse Tensor Accelerators

Each design is described with its own terminologies

Row stationary dataflow
Onchip Bitmask encoding
Zero-aware gating

Eyeriss [JSSC2017]

PlanarTiled-InputStationary-CartesianProduct dataflow
Compressed-sparse-block encoding
Intersection near compute
High-throughput Accumulation

SCNN [ISCA2017]

Hybrid Cartesian product dataflow
Hierarchical encoding
Fast intersection unit with "skipTo" heuristic

ExTensor [MICRO2019]

Row stationary dataflow
CSC encoding
Pipeline that conditionally drop data

Eyeriss V2 [JETCAS2019]

Outer stationary dataflow
Two-level BitMap Encoding
Intersection near compute
Multi-bank Collector


DSTC [ISCA2021]

Row-wise encoding
Processing-in-memory technology

Sparse-ReRAM [ISCA2019]

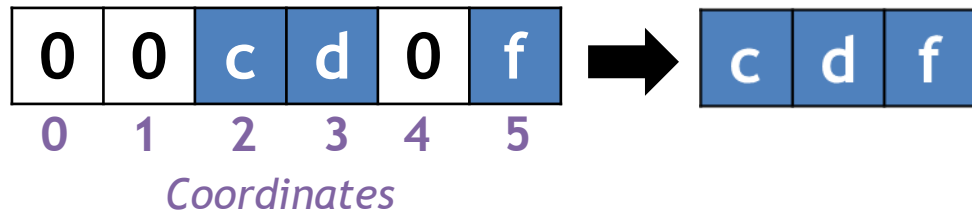
Large, unstructured, and confusing design space

Hardware Sparse Acceleration Features (SAFs)

Format →  ←


Choice of tensor representations

Uncompressed



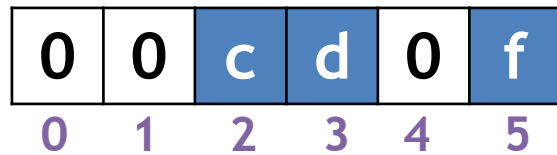
Where exactly was each nonzero in the compressed vector?

Hardware Sparse Acceleration Features (SAFs)

Format →  ←

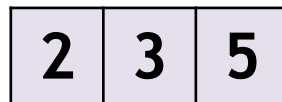
Choice of tensor representations

Uncompressed



Coordinates


Coordinate
Payload



← Metadata



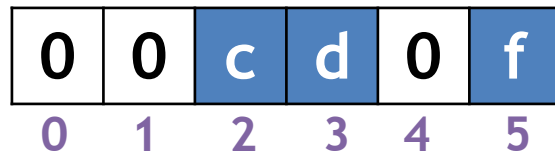
Hardware Sparse Acceleration Features (SAFs)

Format →  ←

Choice of tensor representations

Space Reduction
Energy Reduction

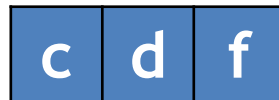
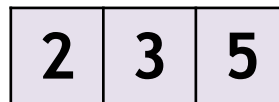
Uncompressed



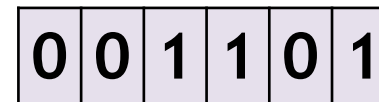
0 1 2 3 4 5

Coordinates

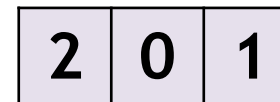
Coordinate
Payload



Bitmask



Run Length
Encoding



Hardware Sparse Acceleration Features (SAFs)

Format →  ←

Choice of tensor representations

Space Reduction
Energy Reduction

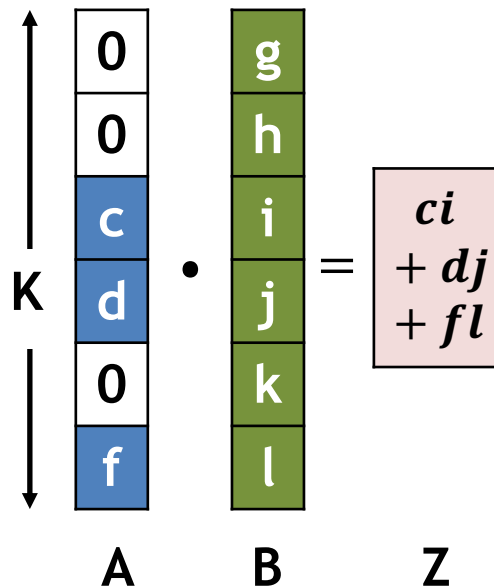
Gating 

Eliminate ineffectual operations by keeping the hardware idle

Example Illustration of Gating's Impact

Example Workload:
Dot Product of Vectors

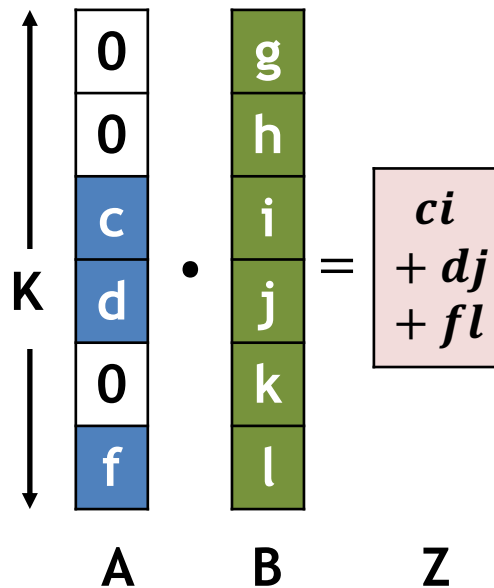
$$Z = \sum_k A_k * B_k$$



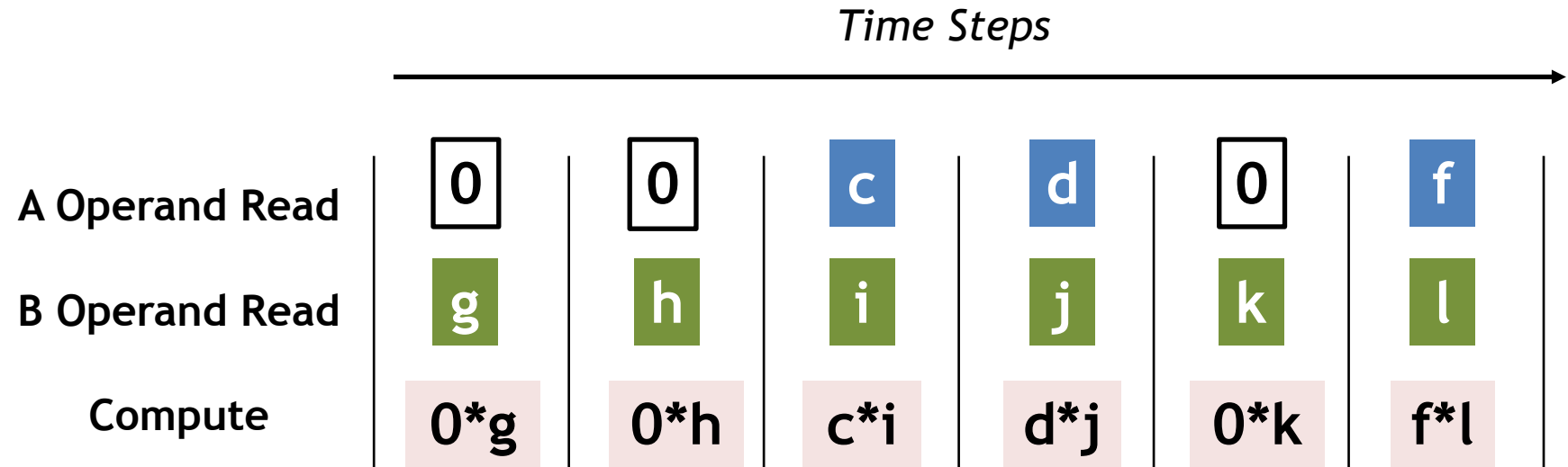
Example Illustration of Gating's Impact

Example Workload:
Dot Product of Vectors

$$Z = \sum_k A_k * B_k$$



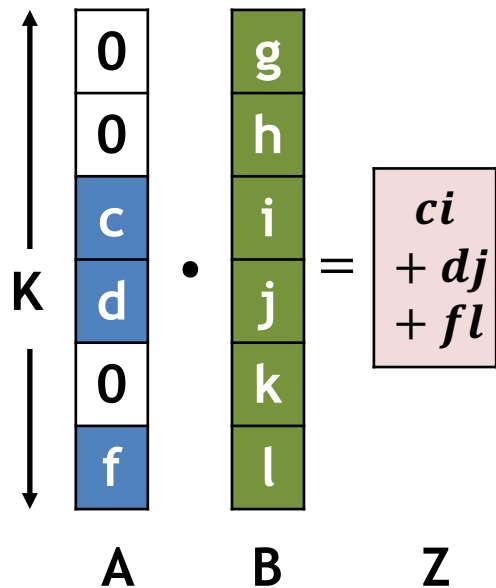
Baseline implementation without any SAFs



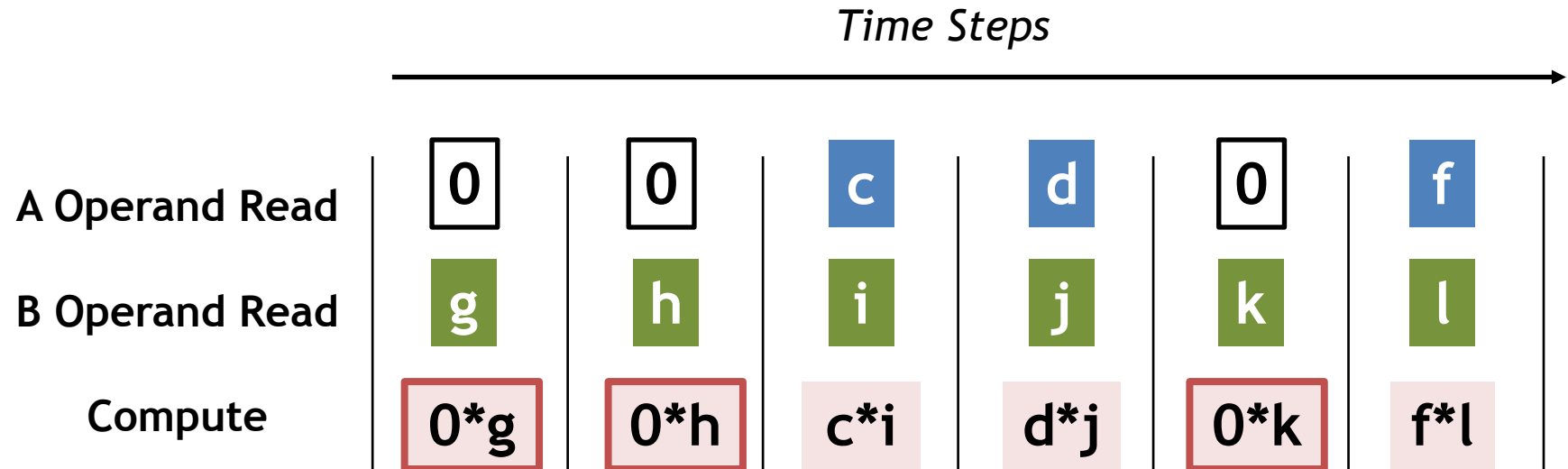
Example Illustration of Gating's Impact

Example Workload:
Dot Product of Vectors

$$Z = \sum_k A_k * B_k$$



Baseline implementation without any SAFs

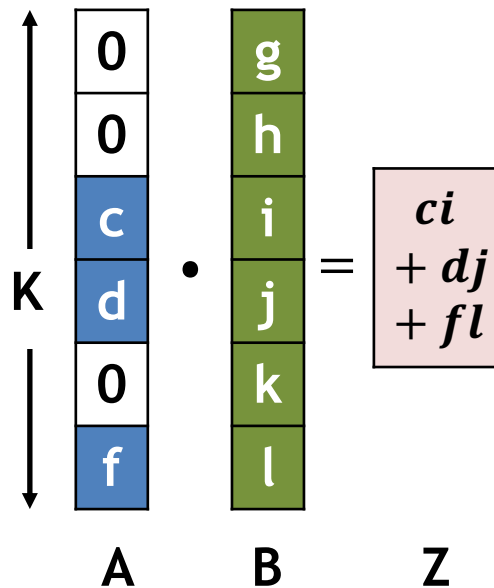


Baseline implementation introduces ineffectual computations

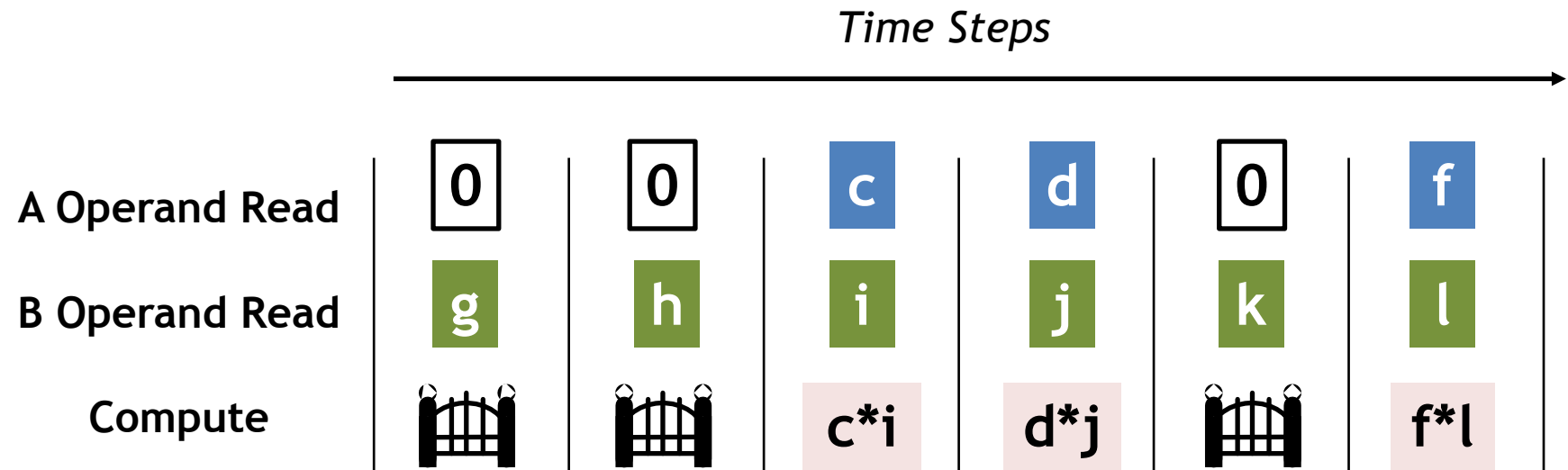
Example Illustration of Gating's Impact

Example Workload:
Dot Product of Vectors

$$Z = \sum_k A_k * B_k$$



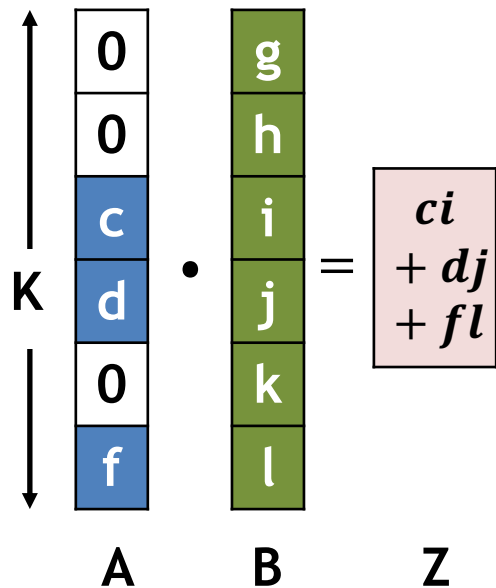
Implementation with
gating SAF applied to compute



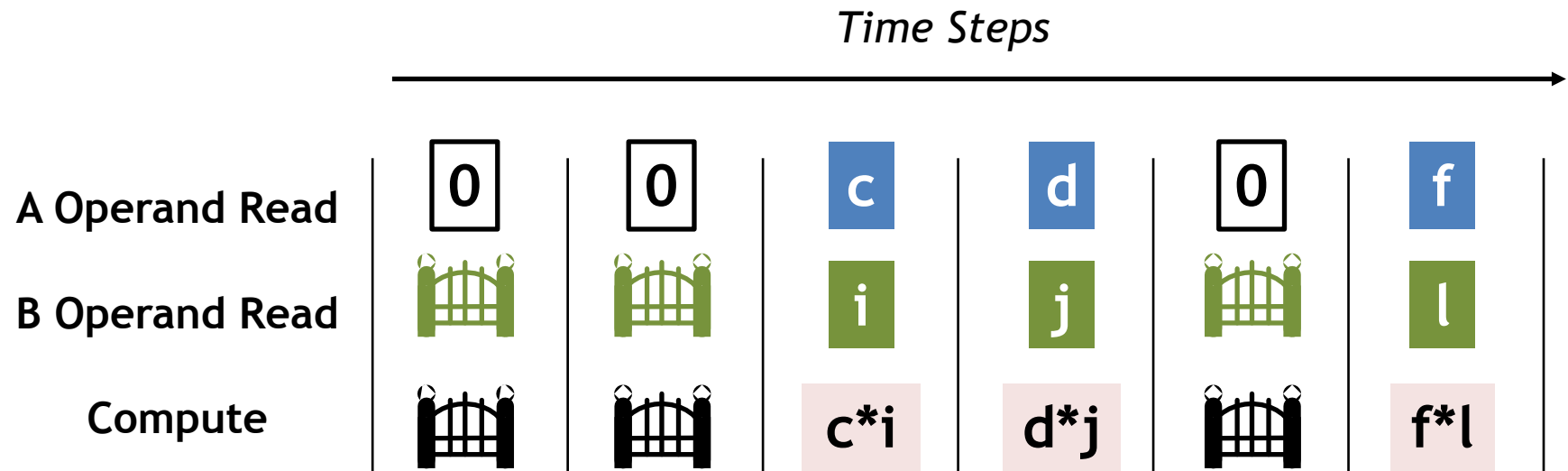
Example Illustration of Gating's Impact

Example Workload:
Dot Product of Vectors

$$Z = \sum_k A_k * B_k$$



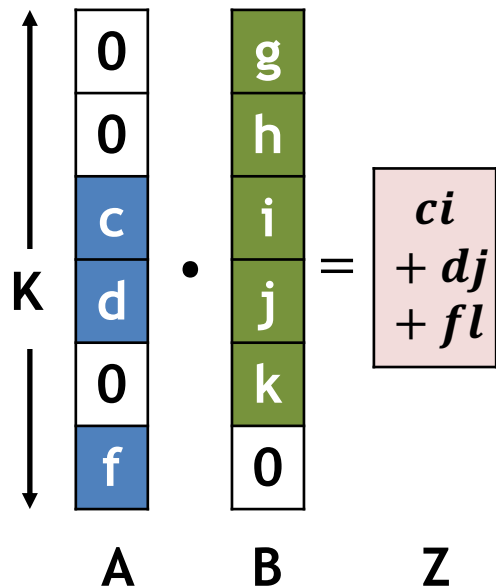
Implementation with
gating SAF applied to B operand read and compute



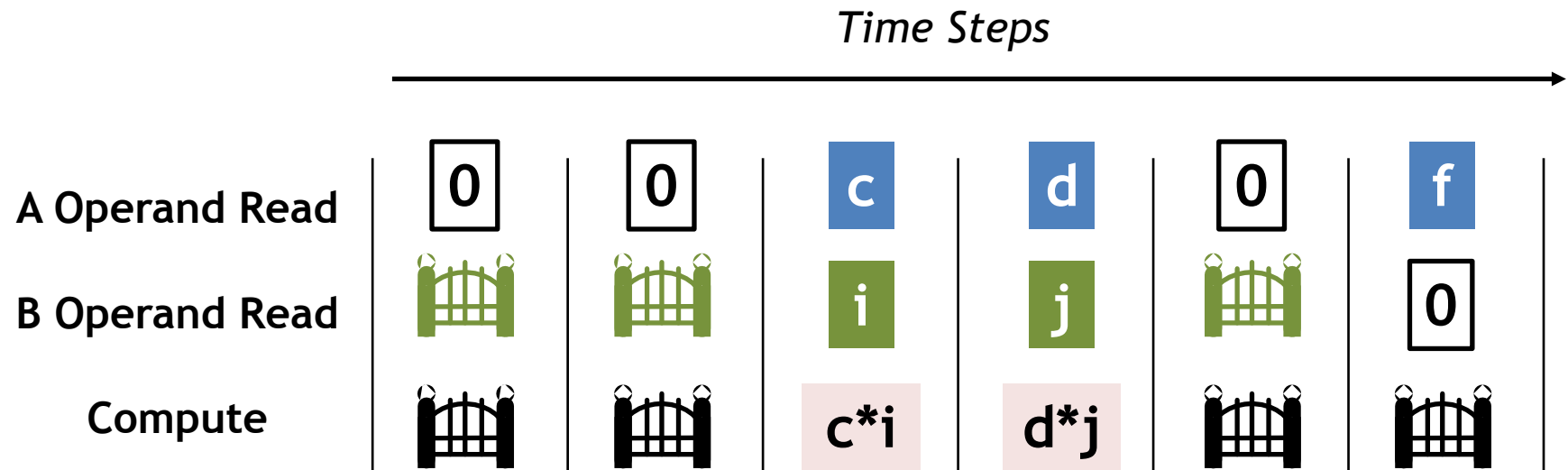
Example Illustration of Gating's Impact

Example Workload:
Dot Product of Vectors

$$Z = \sum_k A_k * B_k$$



Implementation with
gating SAF applied to B operand read and compute



Hardware Sparse Acceleration Features (SAFs)

Format →  ←

Choice of tensor representations

Space Reduction
Energy Reduction

Gating



Eliminate ineffectual operations by keeping the hardware idle

Energy Reduction

Skipping

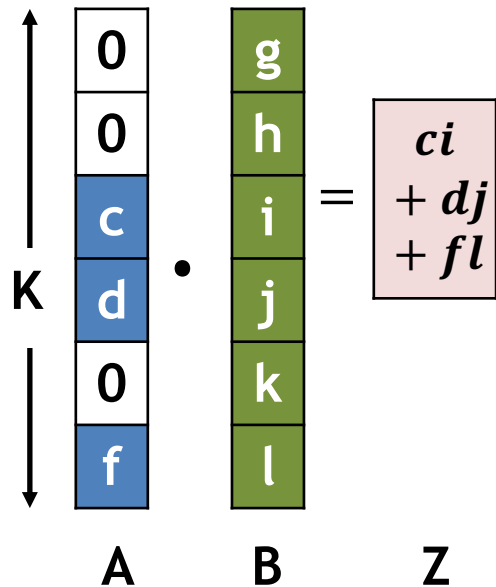


Eliminate ineffectual operations by fast forwarding

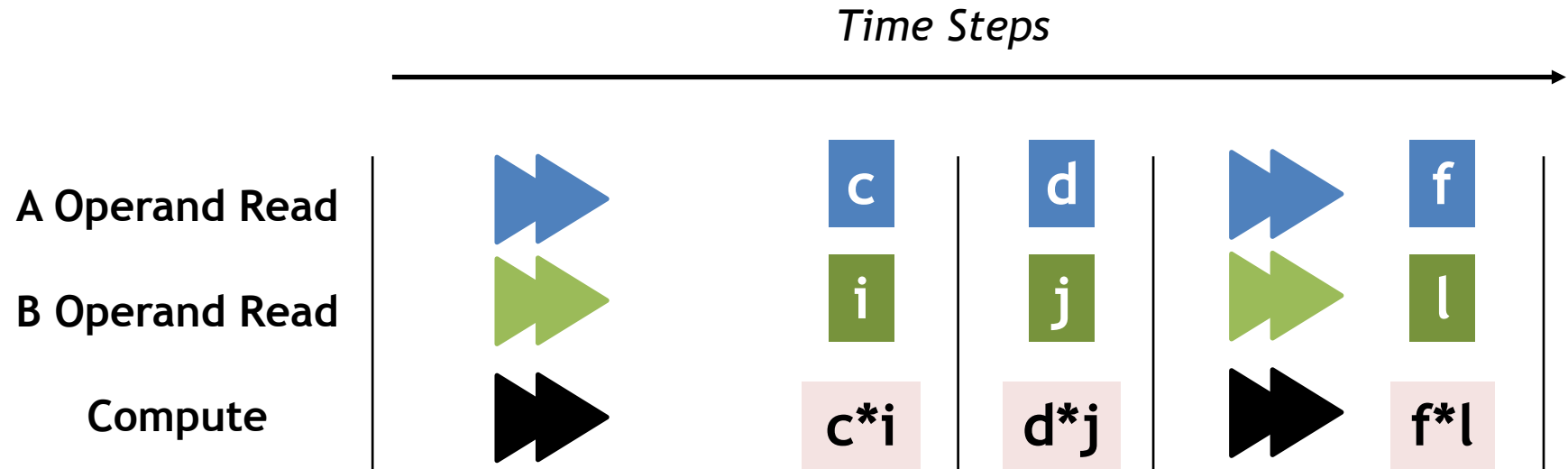
Example Illustration of Skipping's Impact

Example Workload:
Dot Product of Vectors

$$Z = \sum_k A_k * B_k$$




Implementation with
skipping SAF applied to operand reads and compute



Skipping brings additional latency savings

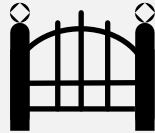
Hardware Sparse Acceleration Features (SAFs)

Format →  ←

Choice tensor representations

Space Reduction
Energy Reduction

Gating



Eliminate ineffectual operations by keeping the hardware idle

Energy Reduction

Skipping

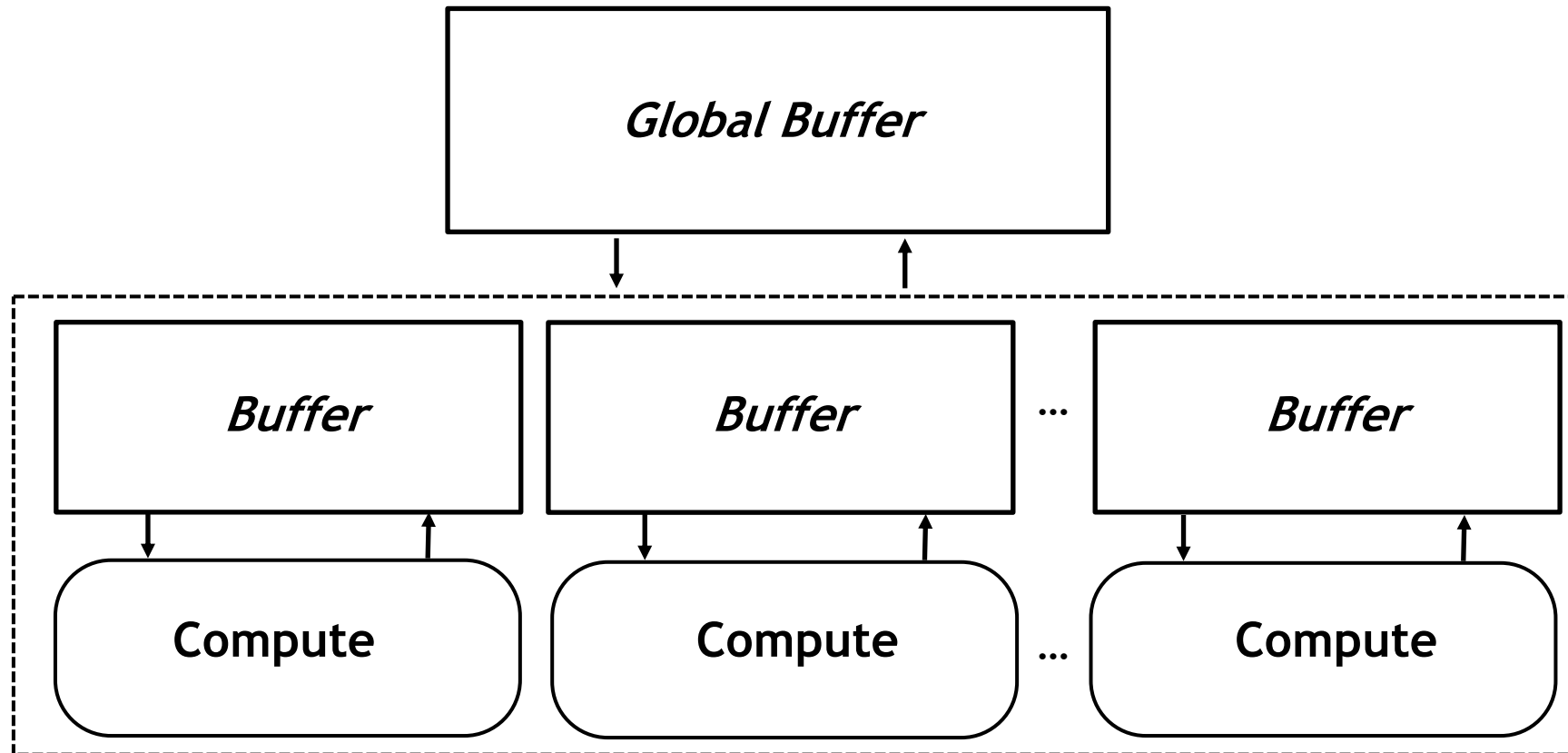


Eliminate ineffectual operations by fast forwarding

Energy Reduction
Latency Reduction

Systematic Descriptions of Accelerators

Proposed Approach:
assigning SAFs to each level in the architecture

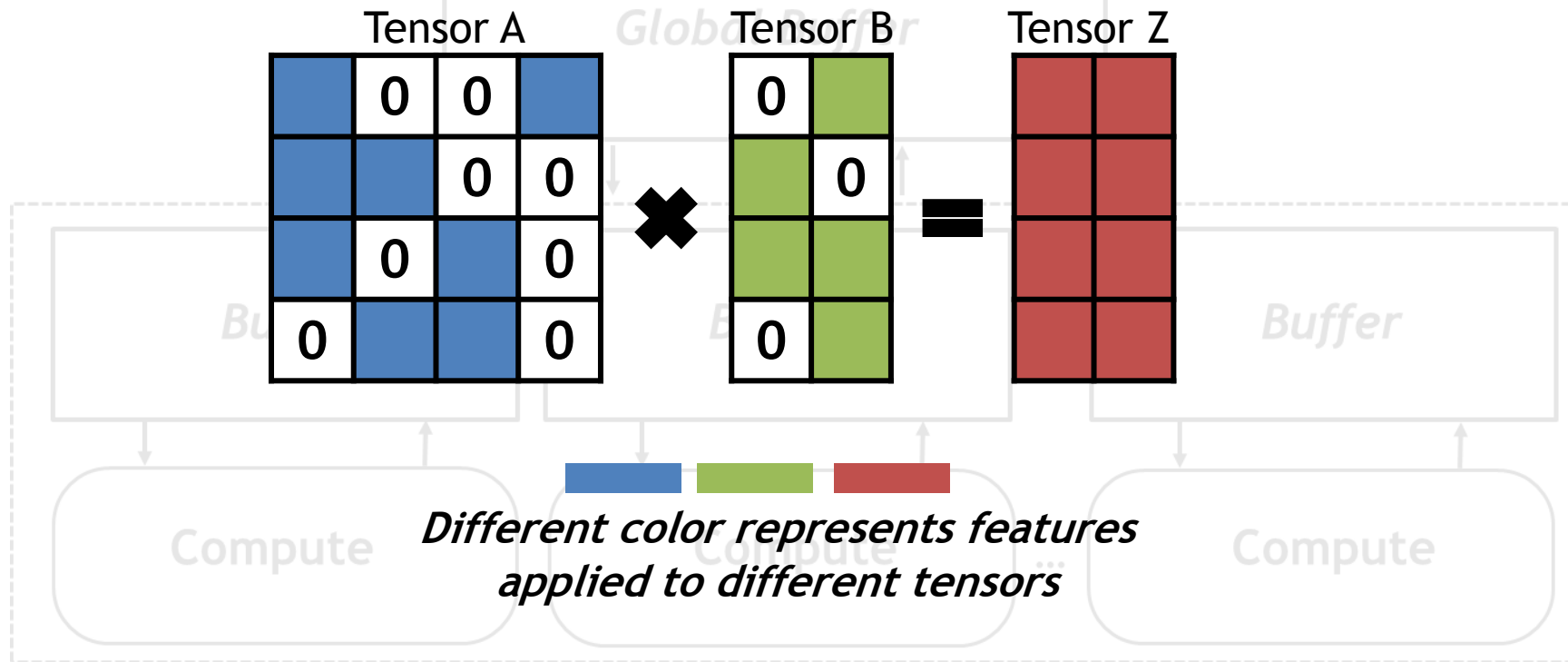


Example Accelerator Architecture Organization

Systematic Descriptions of Accelerators

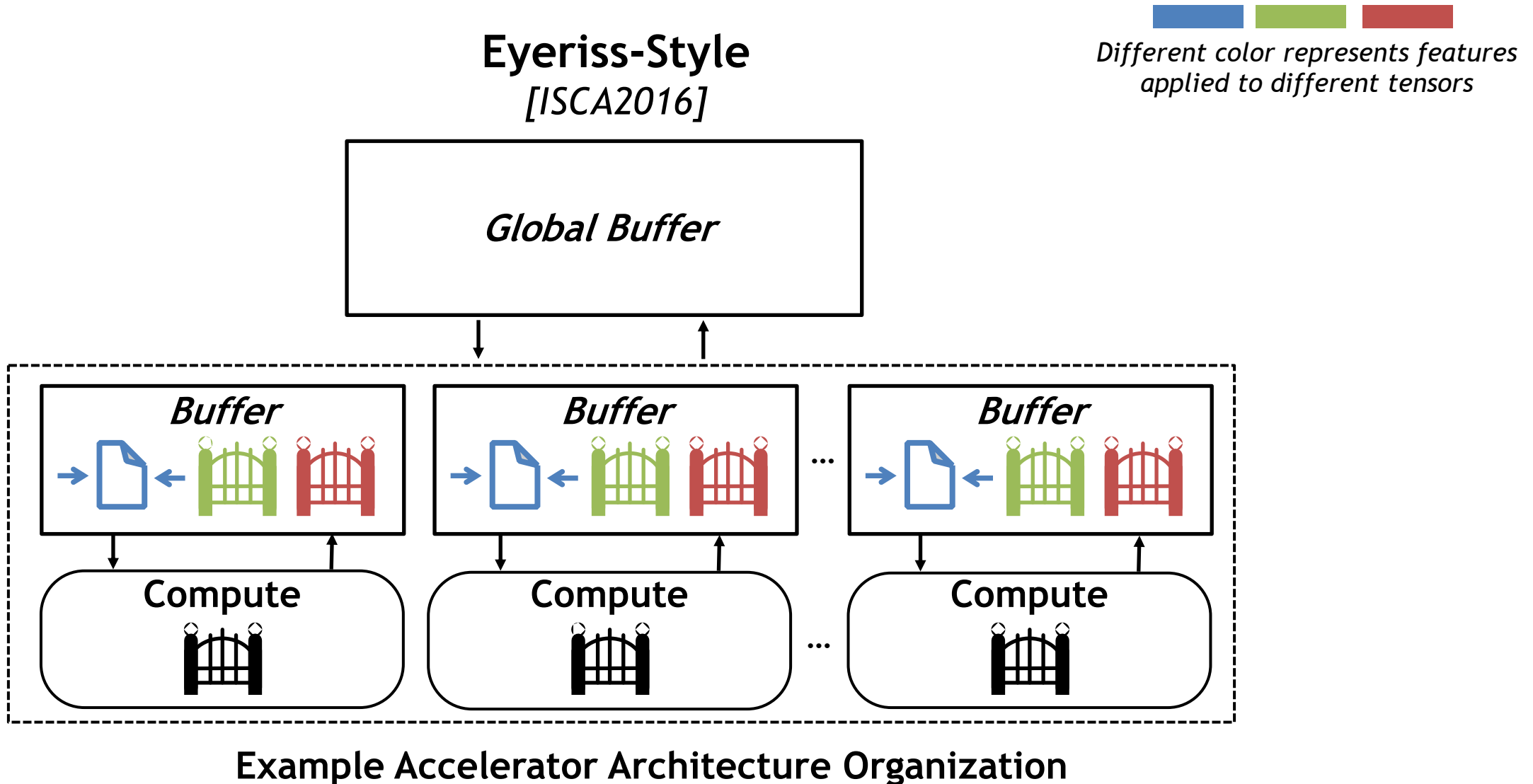
Sparsity-related optimizations can be described by assigning SAFs to each level in the architecture

SAFs can be associated with different tensors in the workload

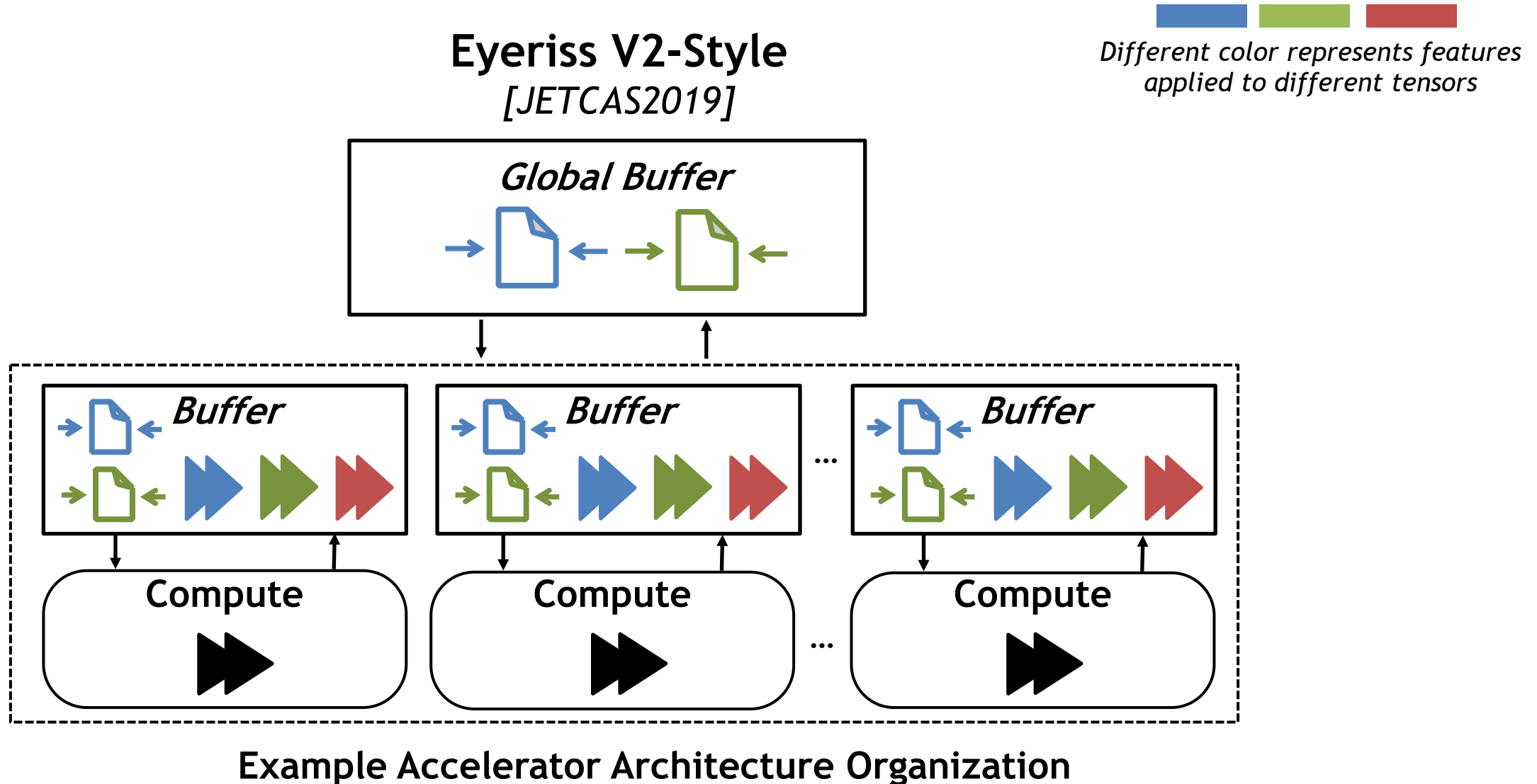


Example Accelerator Architecture Organization

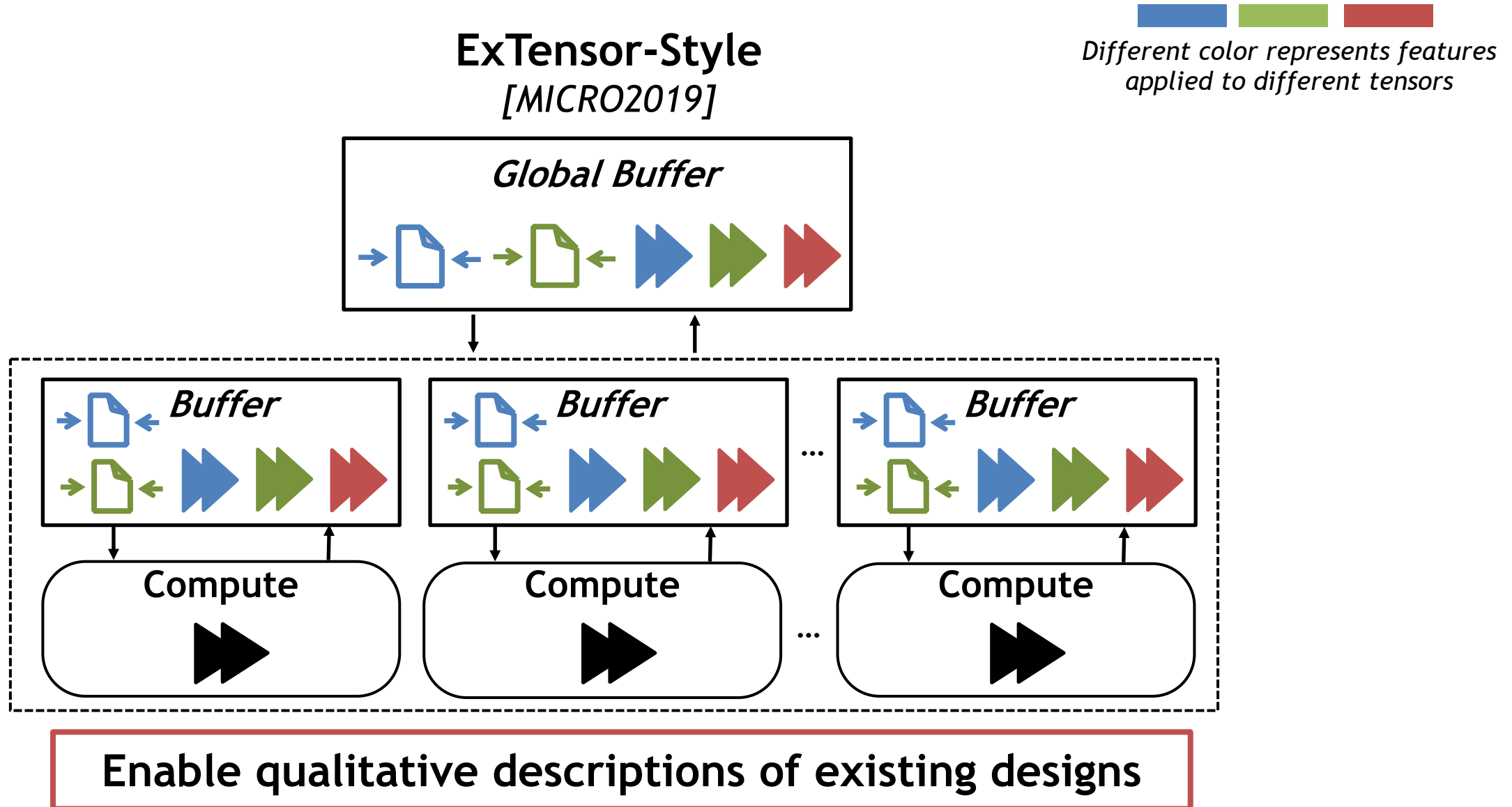
Systematic Descriptions of Accelerators



Systematic Descriptions of Accelerators



Systematic Descriptions of Accelerators





We Need a Quantitative Modeling Framework



Fast



Accurate



Flexible



Existing Modeling Frameworks are Insufficient

(Design-Specific) Cycle-Level Simulators

*SCNN[ISCA16], STONNE[CAL21],
MAGNET[ICCAD19], DNNBuilder[ICCAD18], etc.*

Slow



Inflexible



General Analytical Modeling Frameworks

*Timeloop[ISPASS19], MAESTRO[MICRO19],
Scale-Sim[ISPASS20], CoSA[ISCA21], etc.*

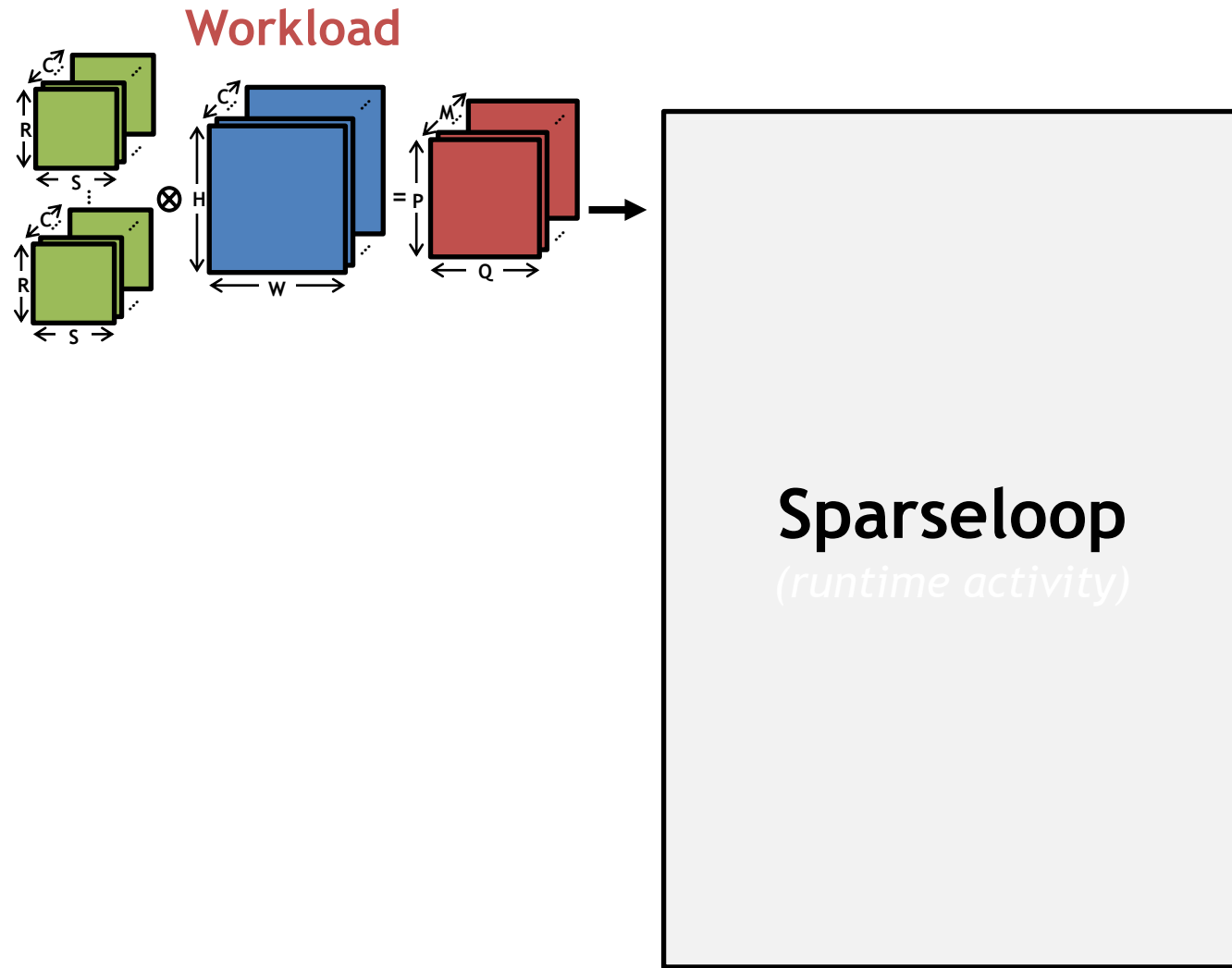
No Sparsity Support



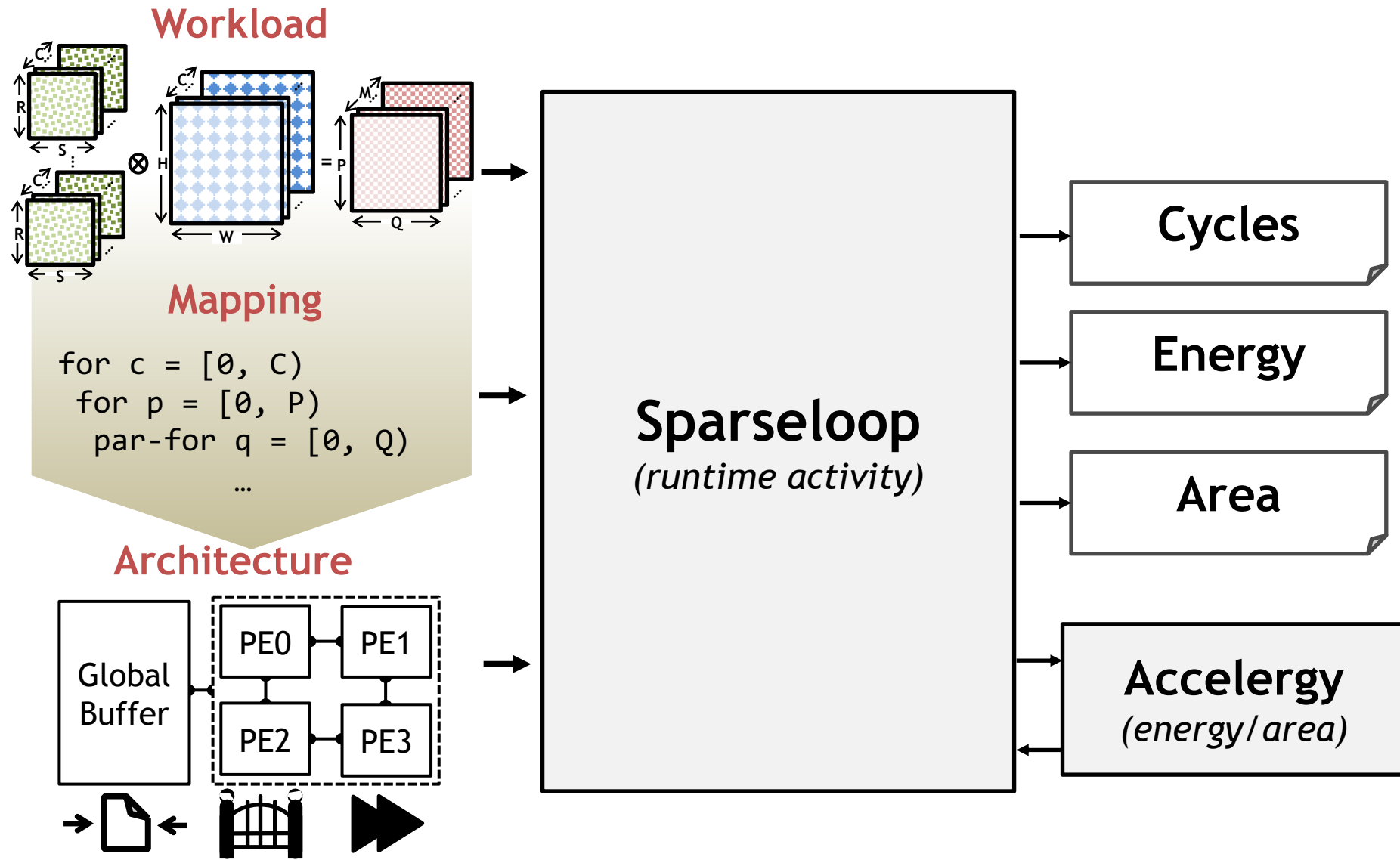
II. Sparseloop: Early-Design Stage Modeling of Sparse Tensor Accelerators

- Statistical Workload Characterization
- Progressive Modeling Procedure
- Modularized SAF Impact Analysis
- Experimental Results

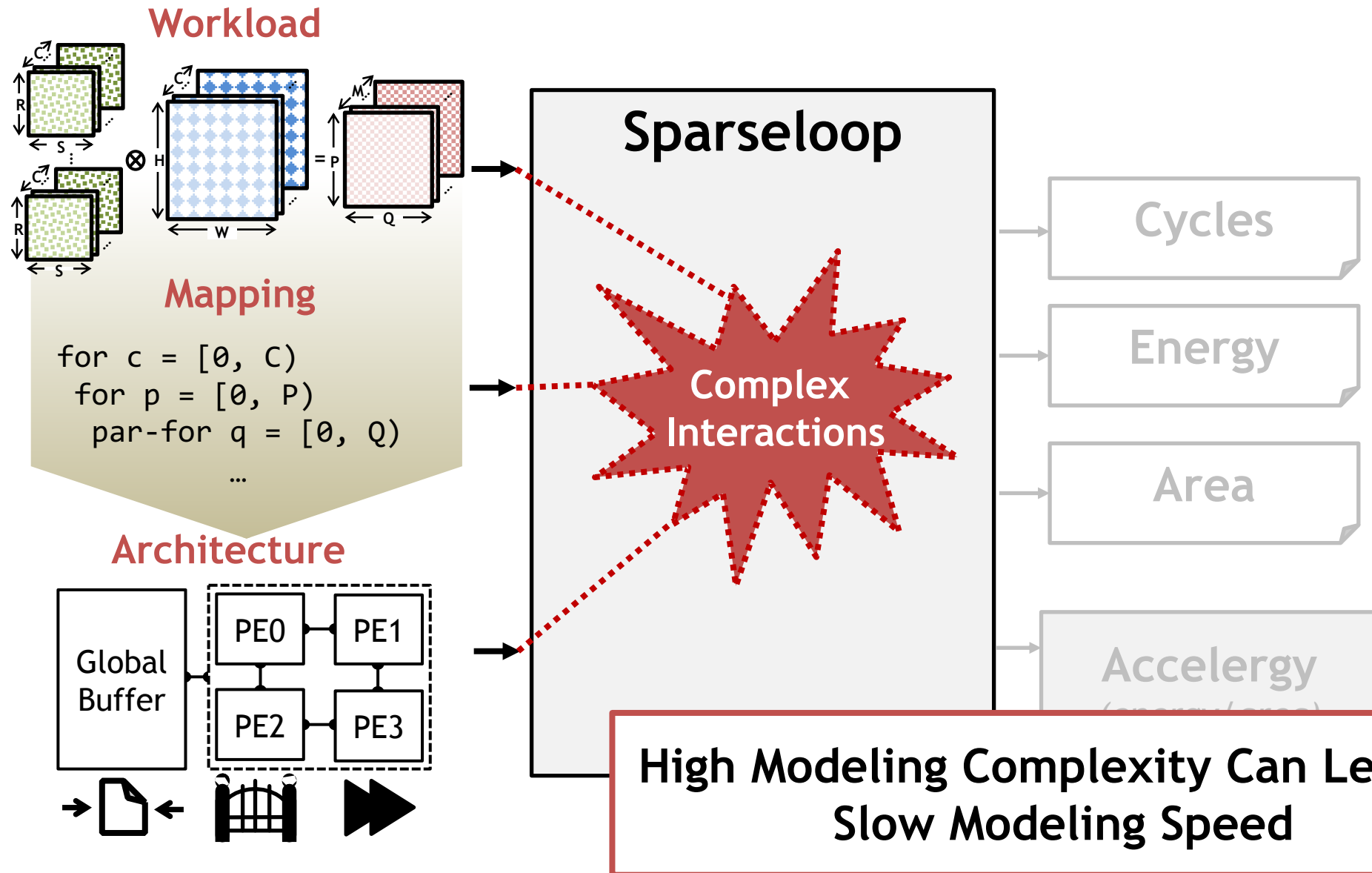
Sparseloop High-Level Framework



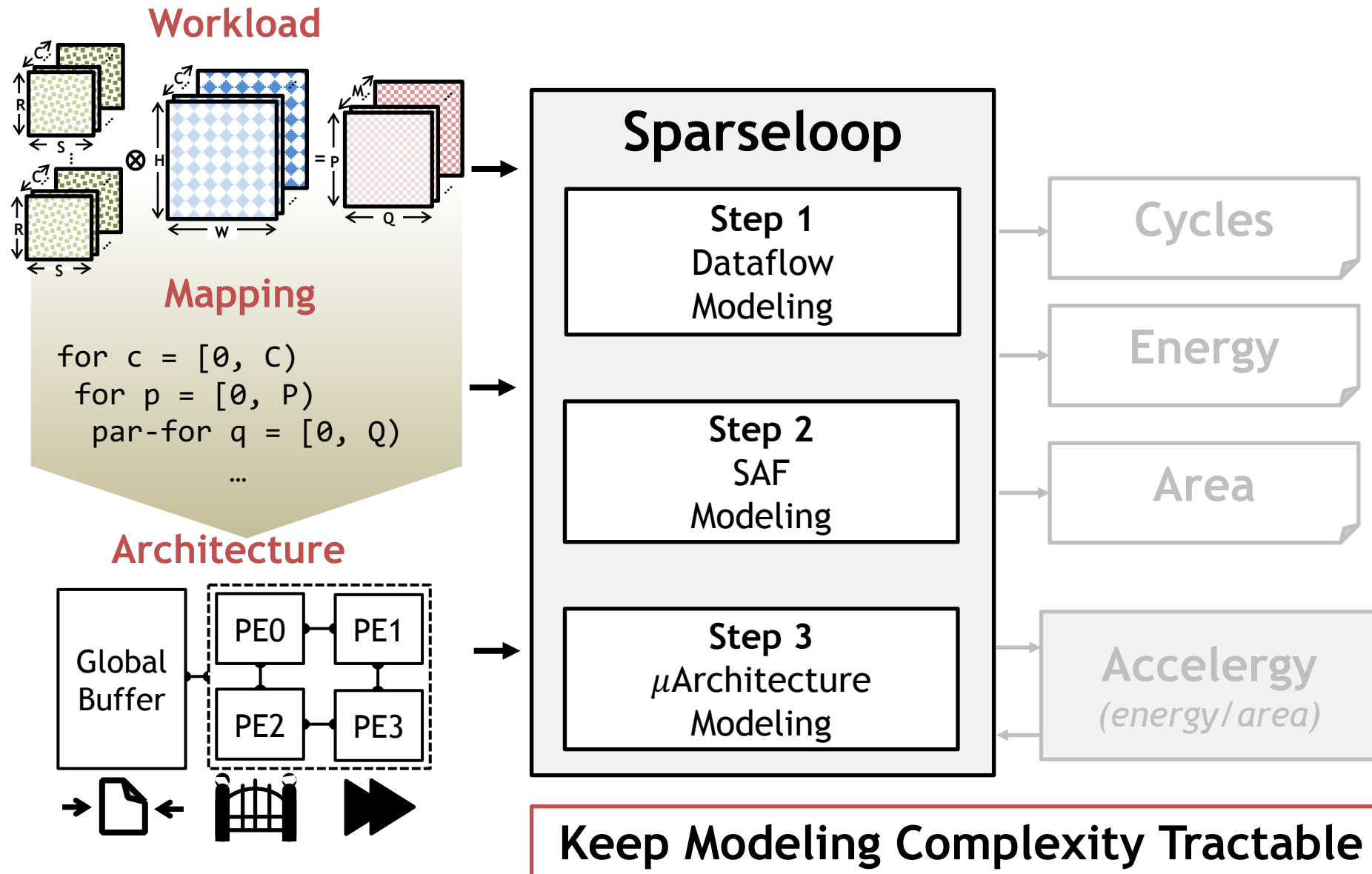
Sparseloop High-Level Framework



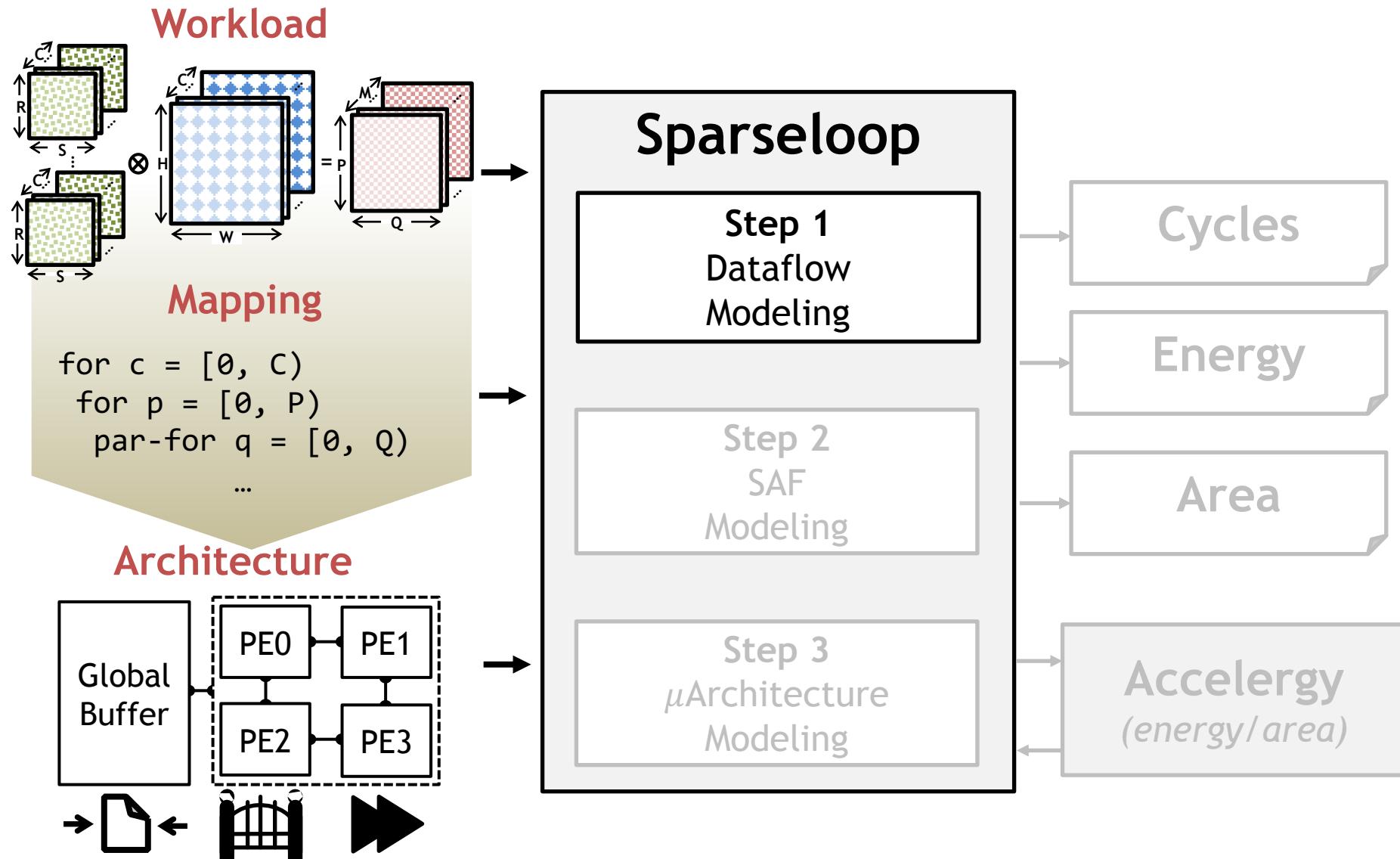
Complex Interactions Between Inputs



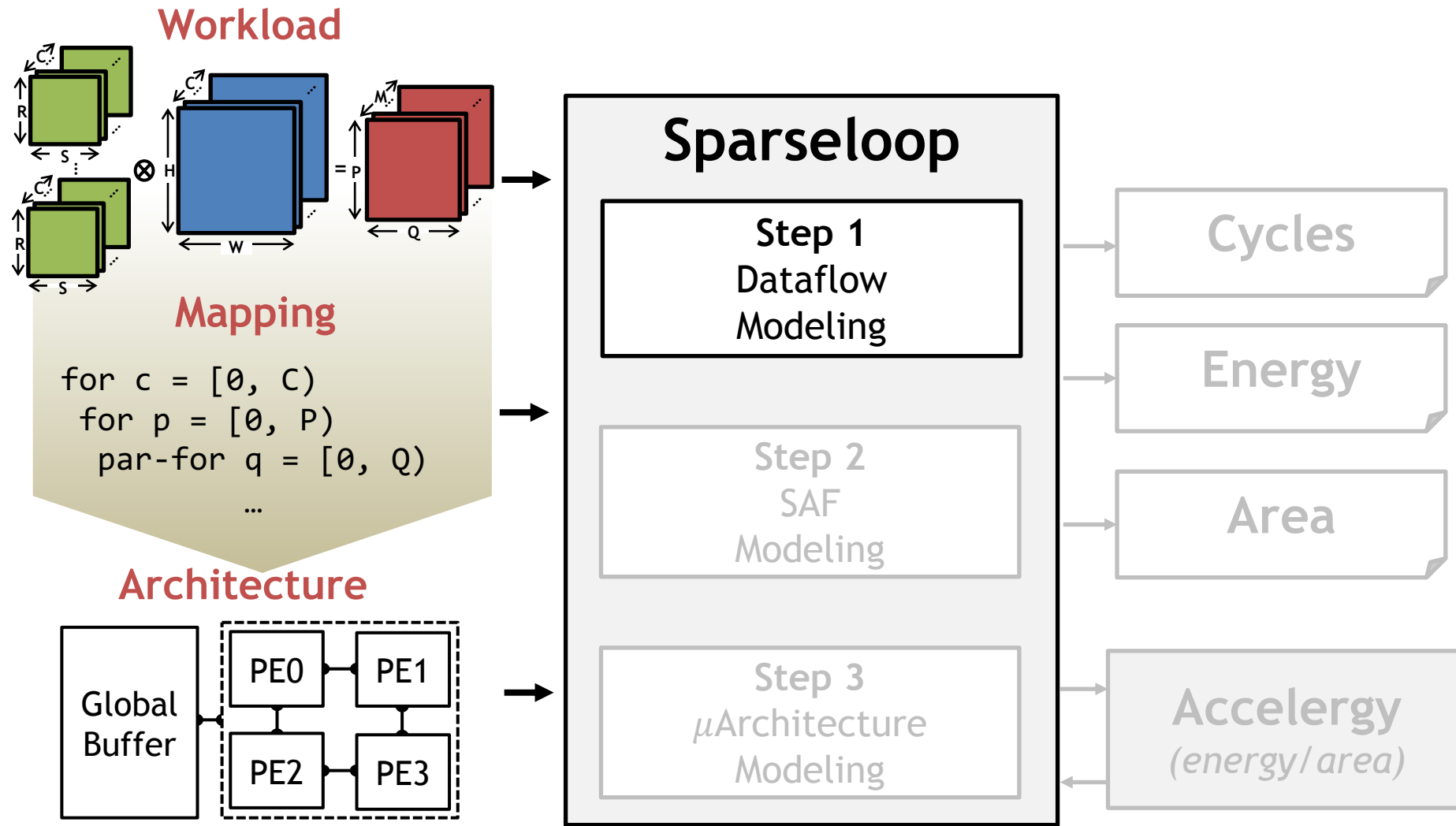
Progressive Modeling Procedure



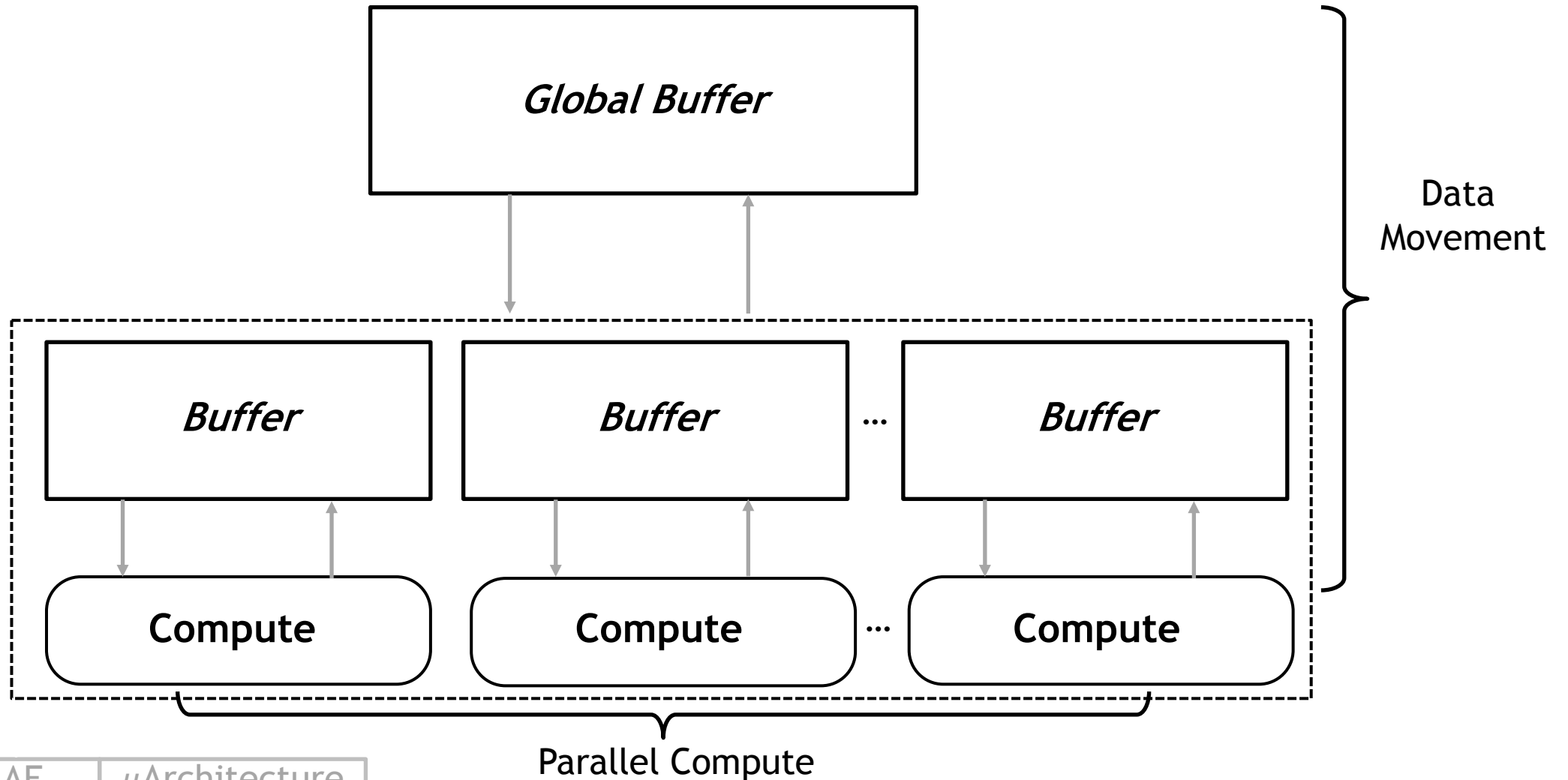
Dataflow Modeling Does Not Consider Sparsity



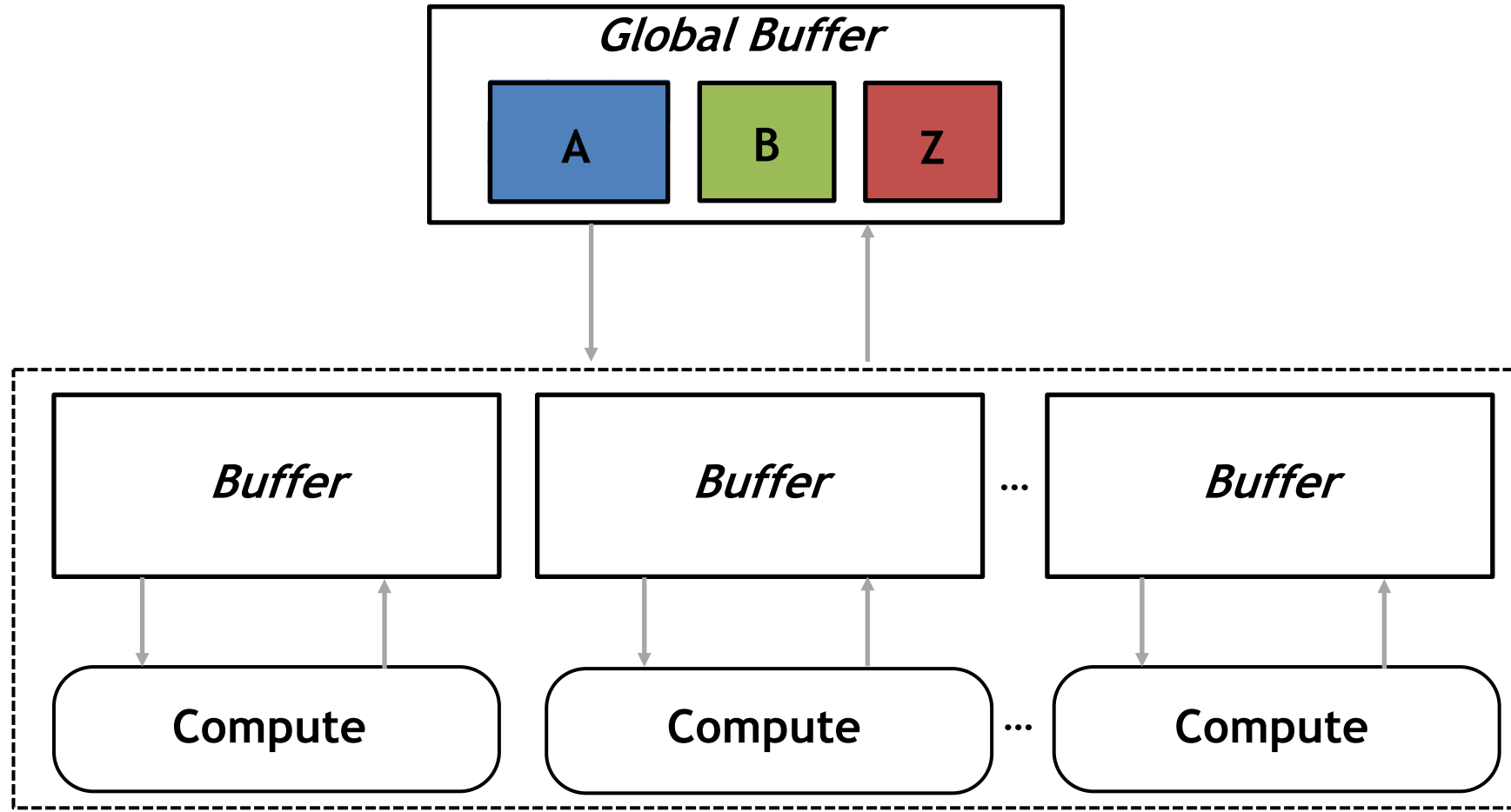
Dataflow Modeling Does Not Consider Sparsity



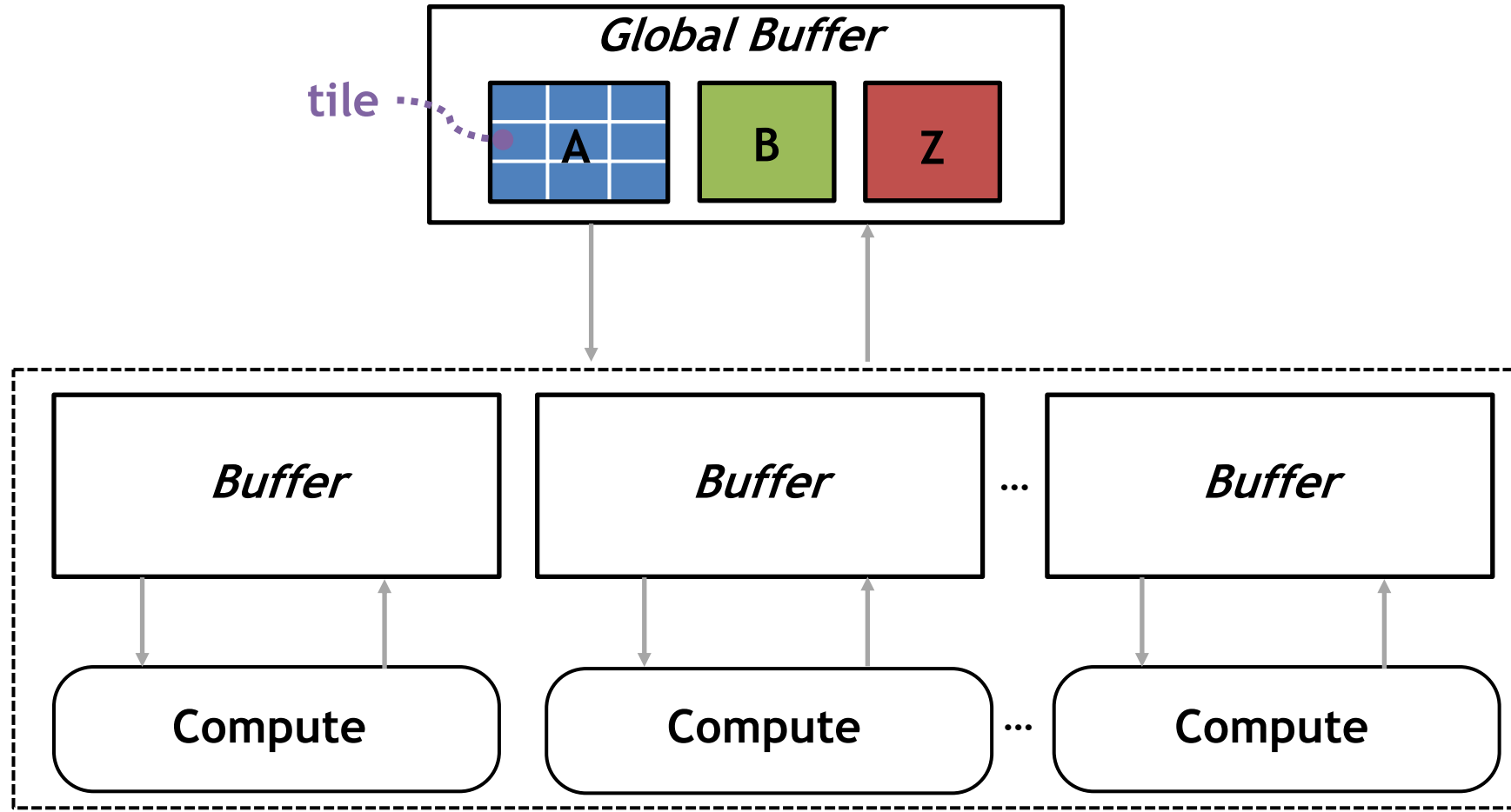
Uncompressed Data Movement and Dense Compute



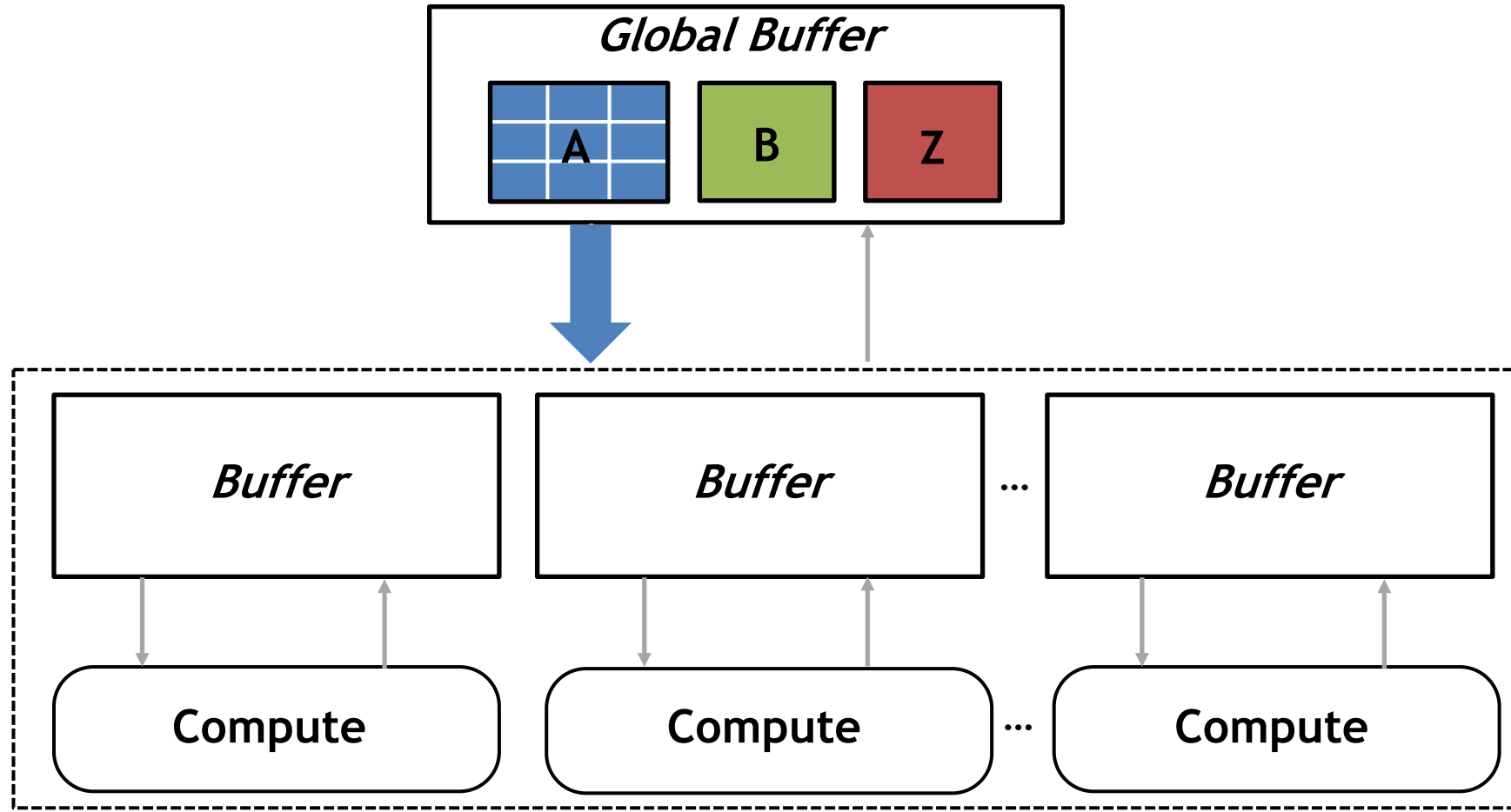
Uncompressed Data Movement and Dense Compute



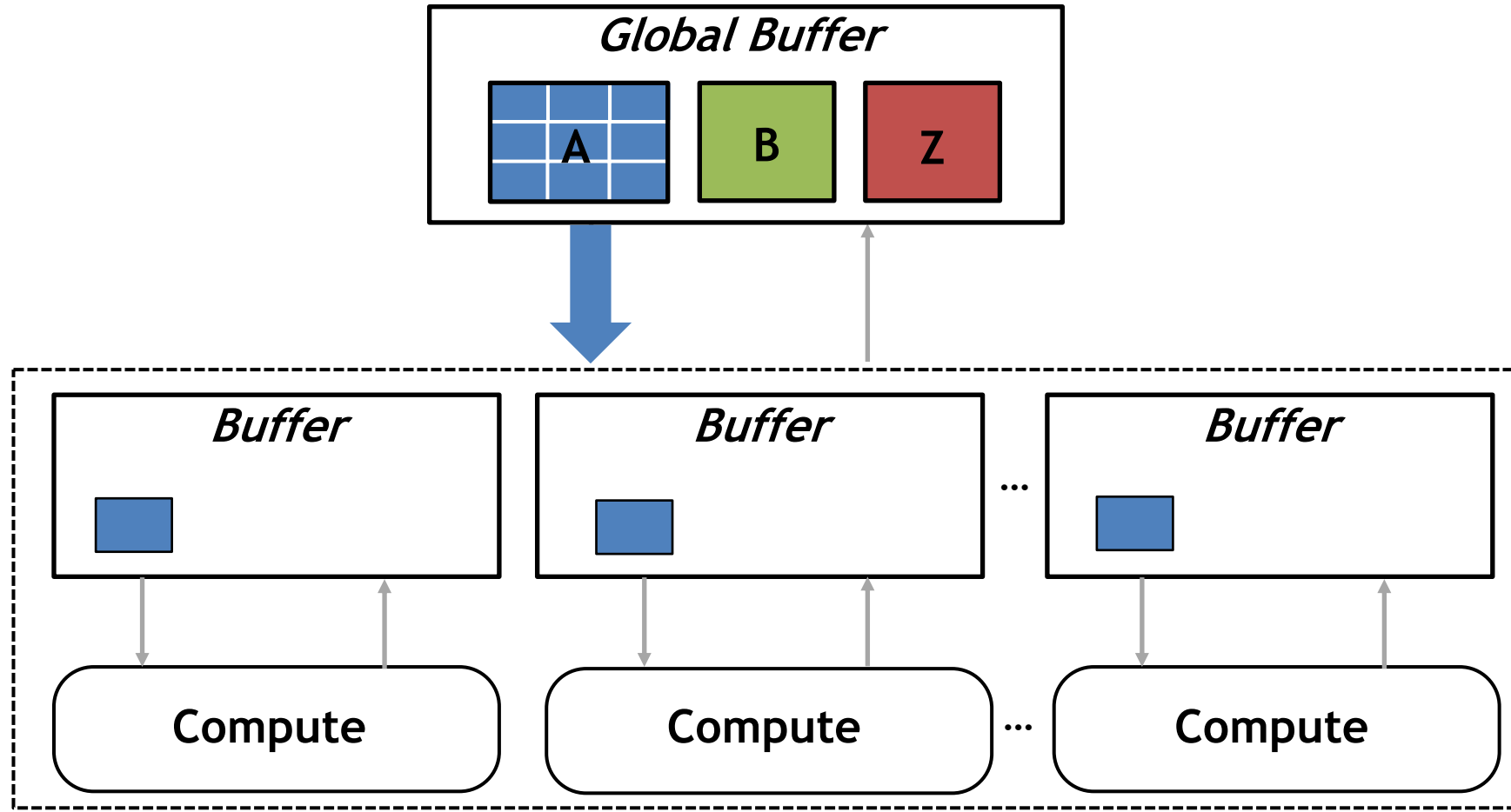
Tensors are Tiled and Transferred Between Levels



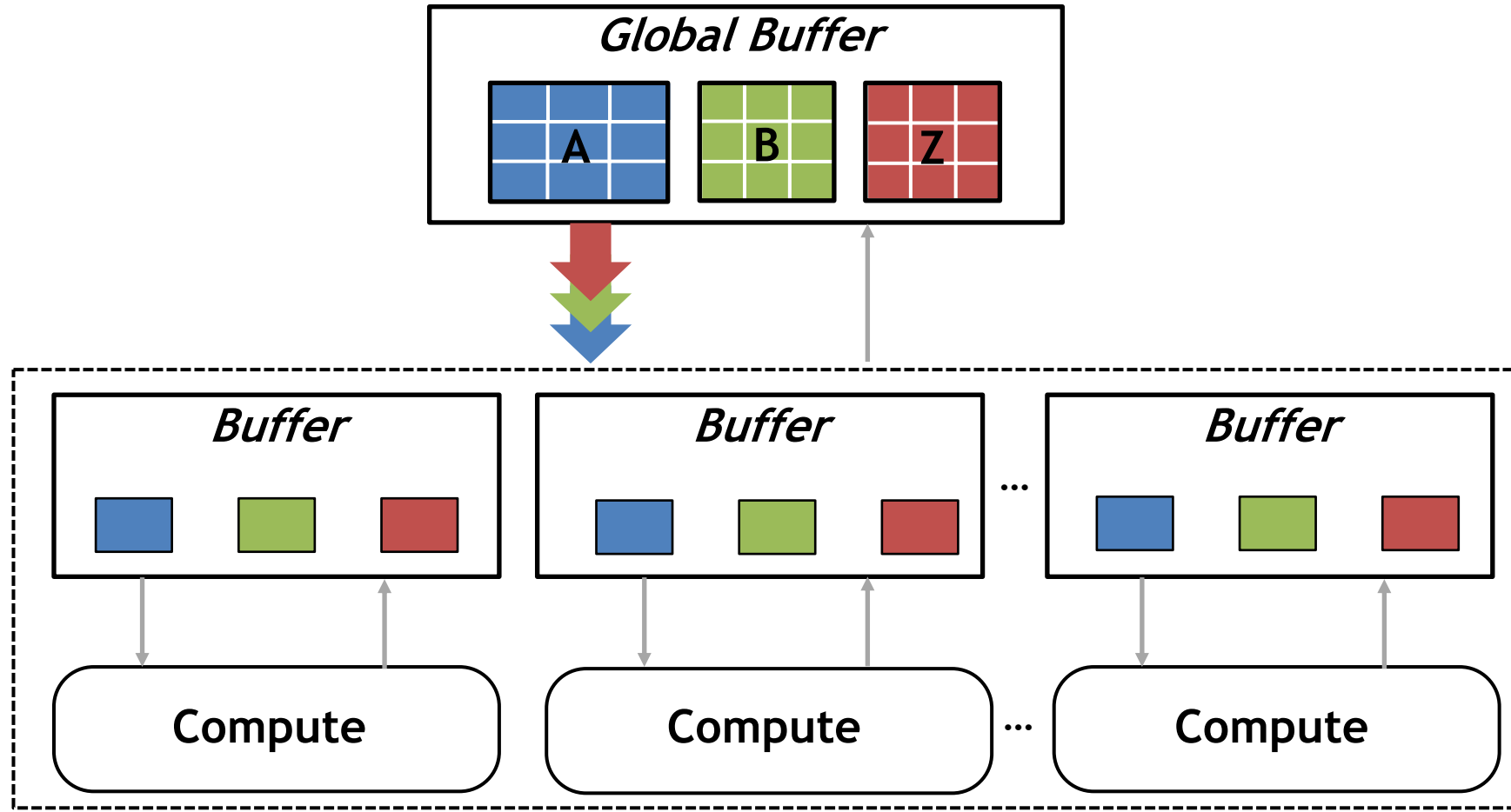
Tensors are Tiled and Transferred Between Levels



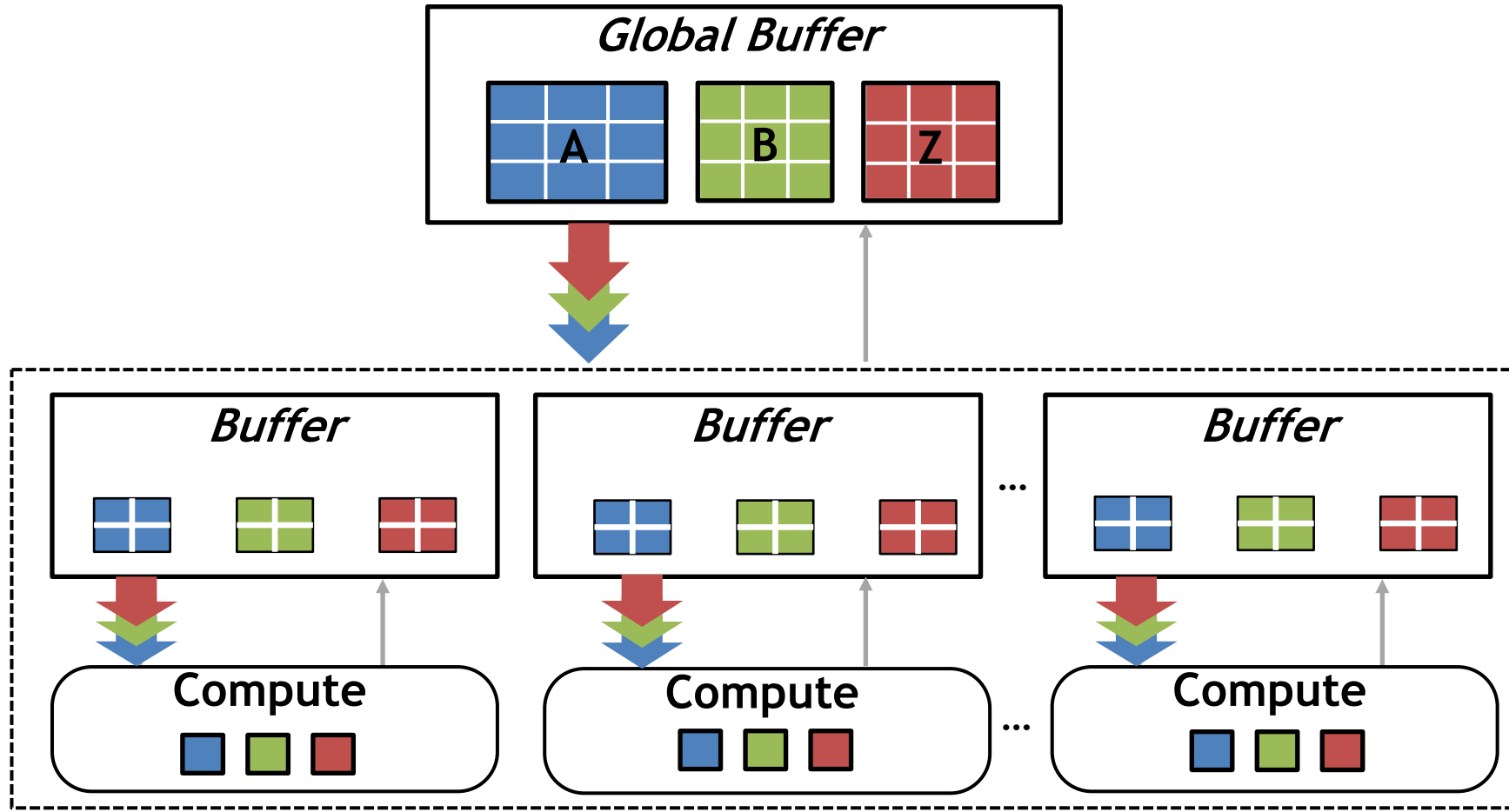
Tensors are Tiled and Transferred Between Levels



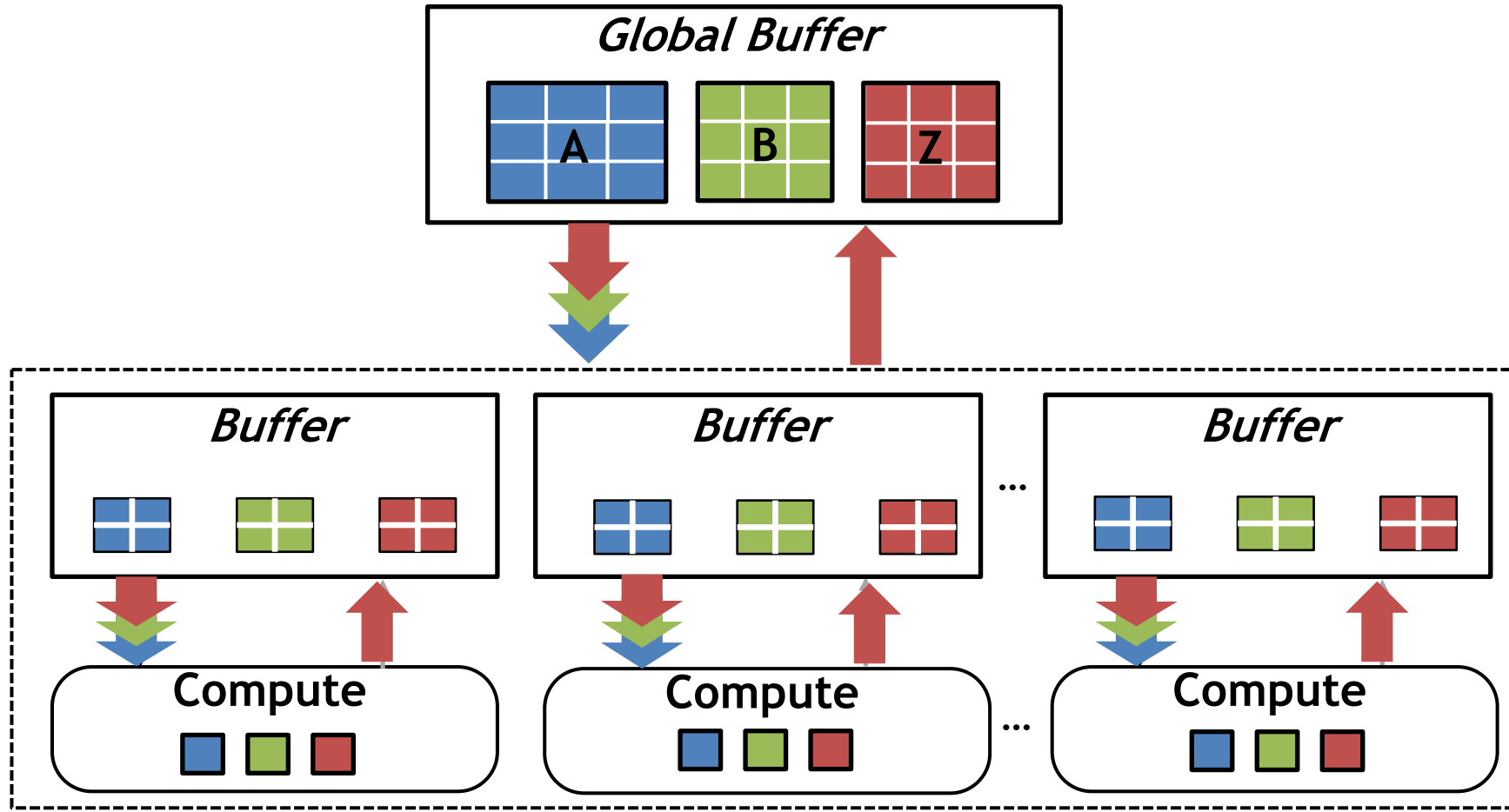
Tensors are Tiled and Transferred Between Levels



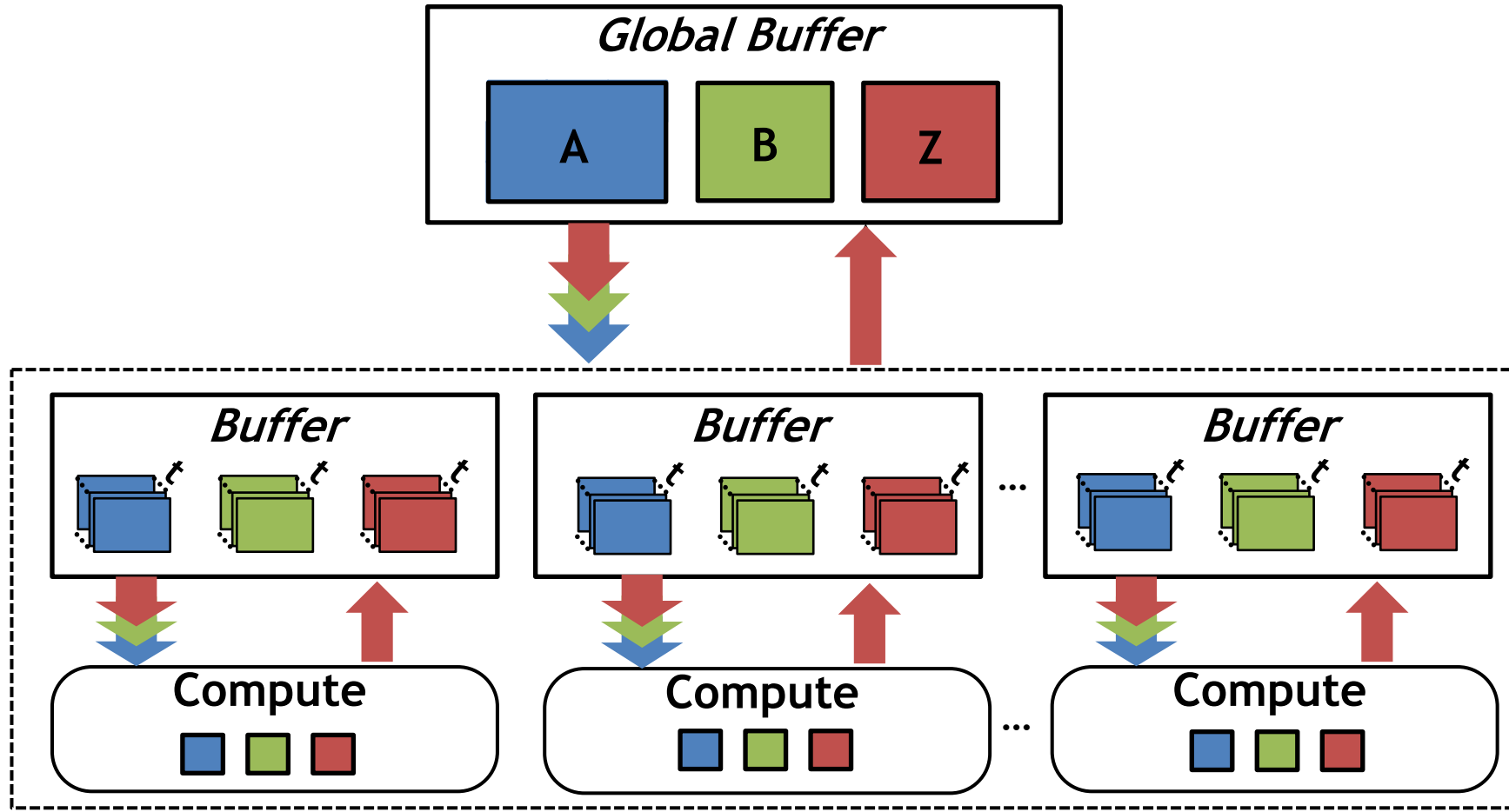
Parallel Computation at Compute Units



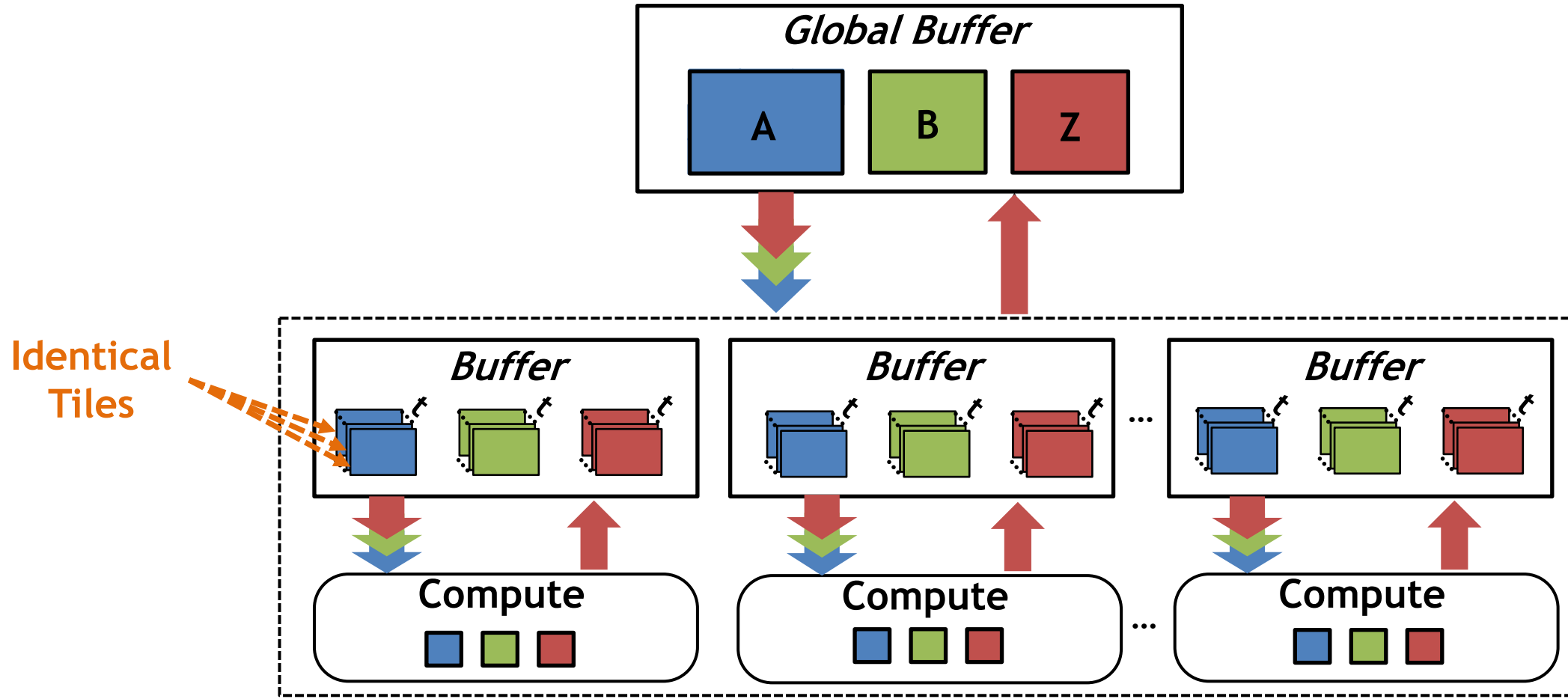
Parallel Computation at Compute Units



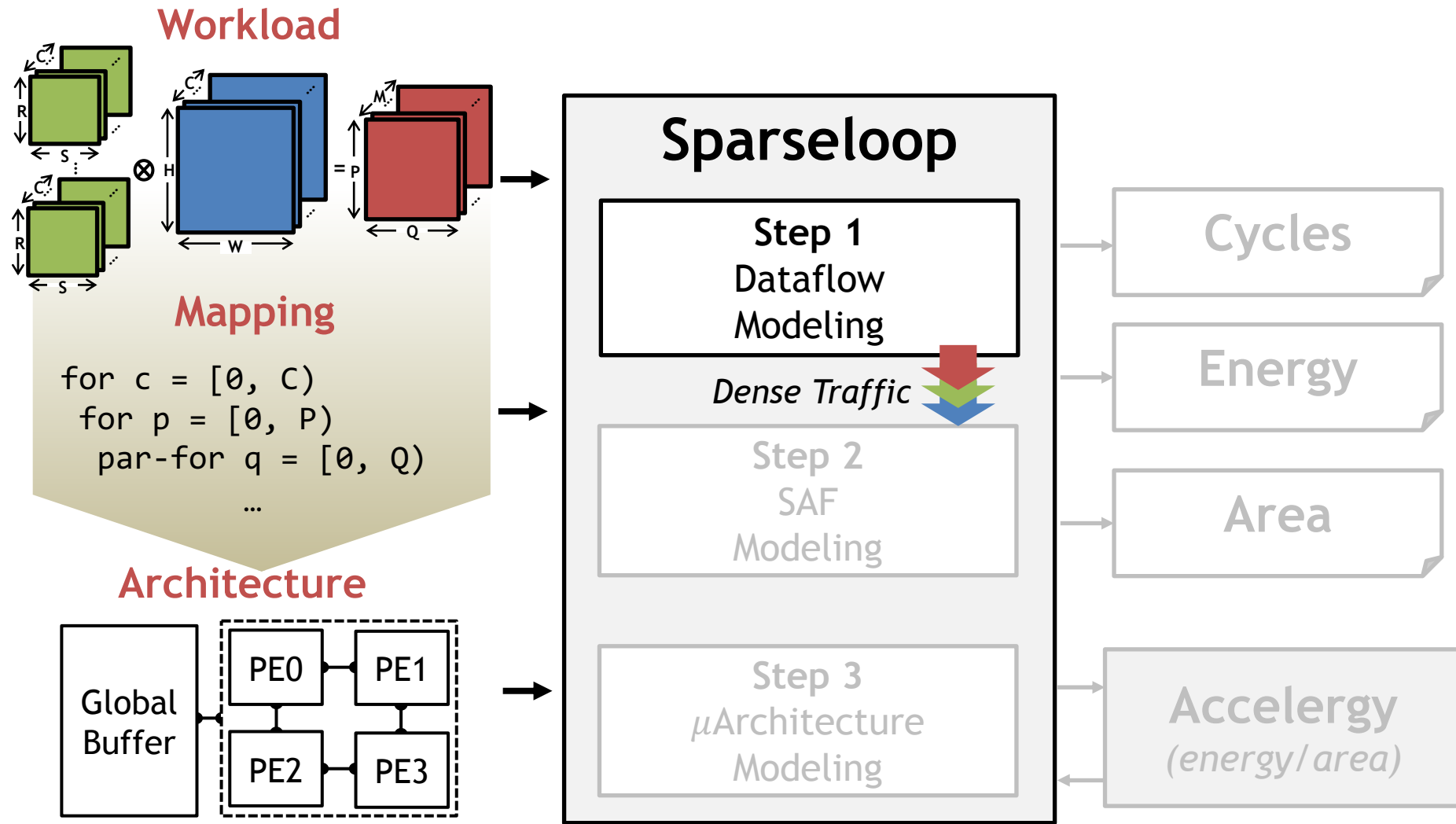
Different Tiles are Transferred Overtime



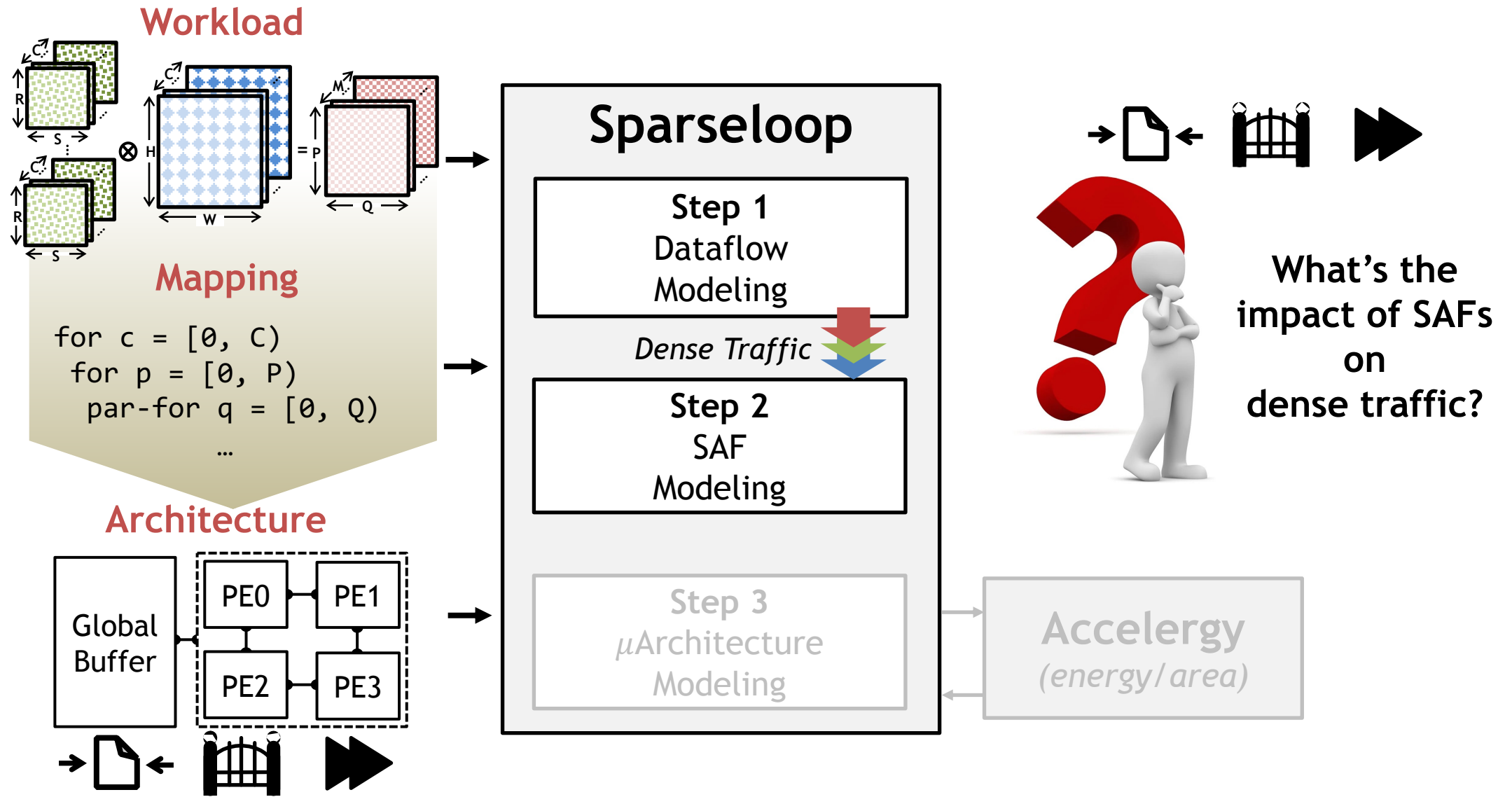
Runtime Activities Can be Quickly Derived



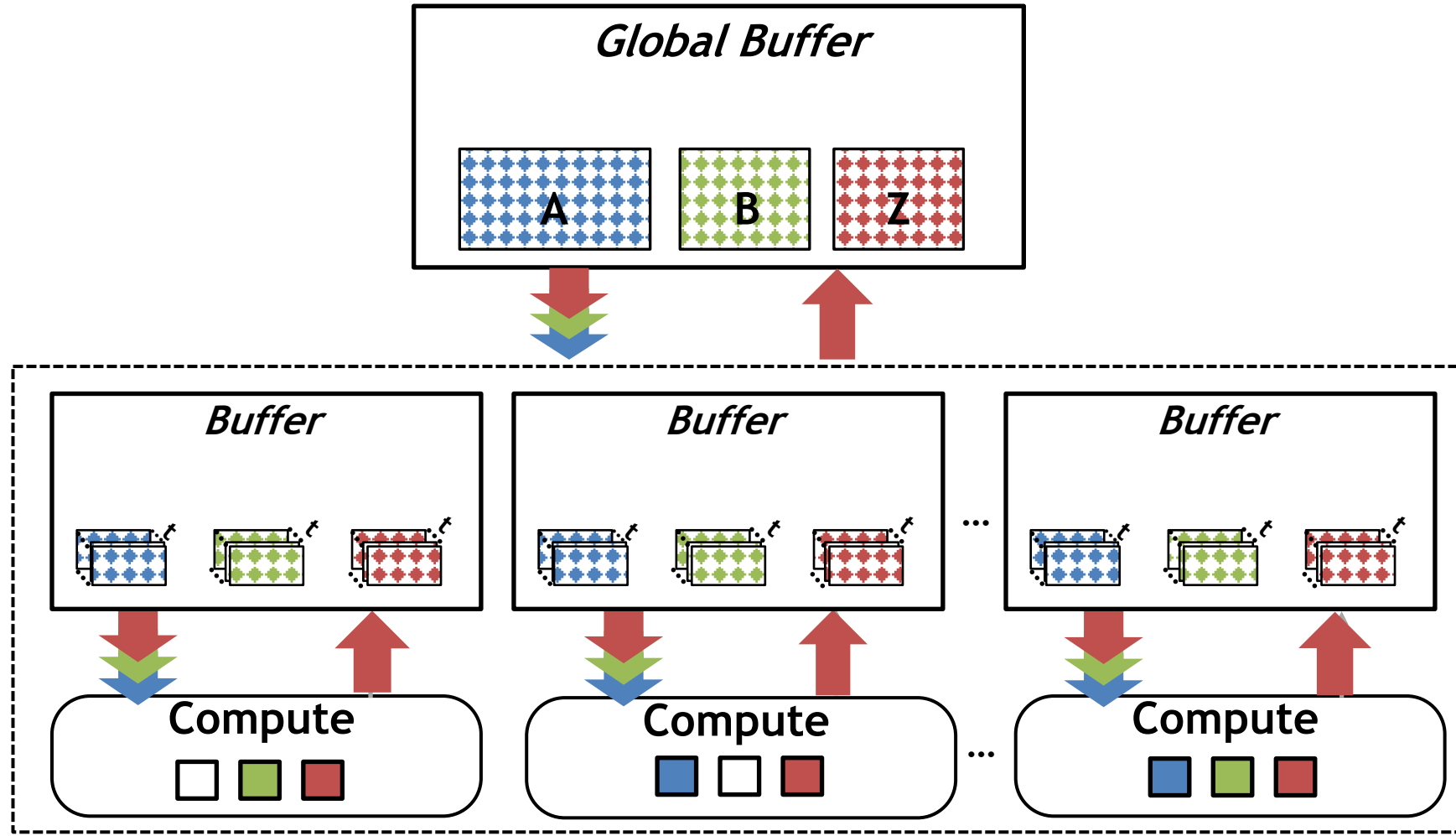
Dataflow Modeling Produces Dense Traffic



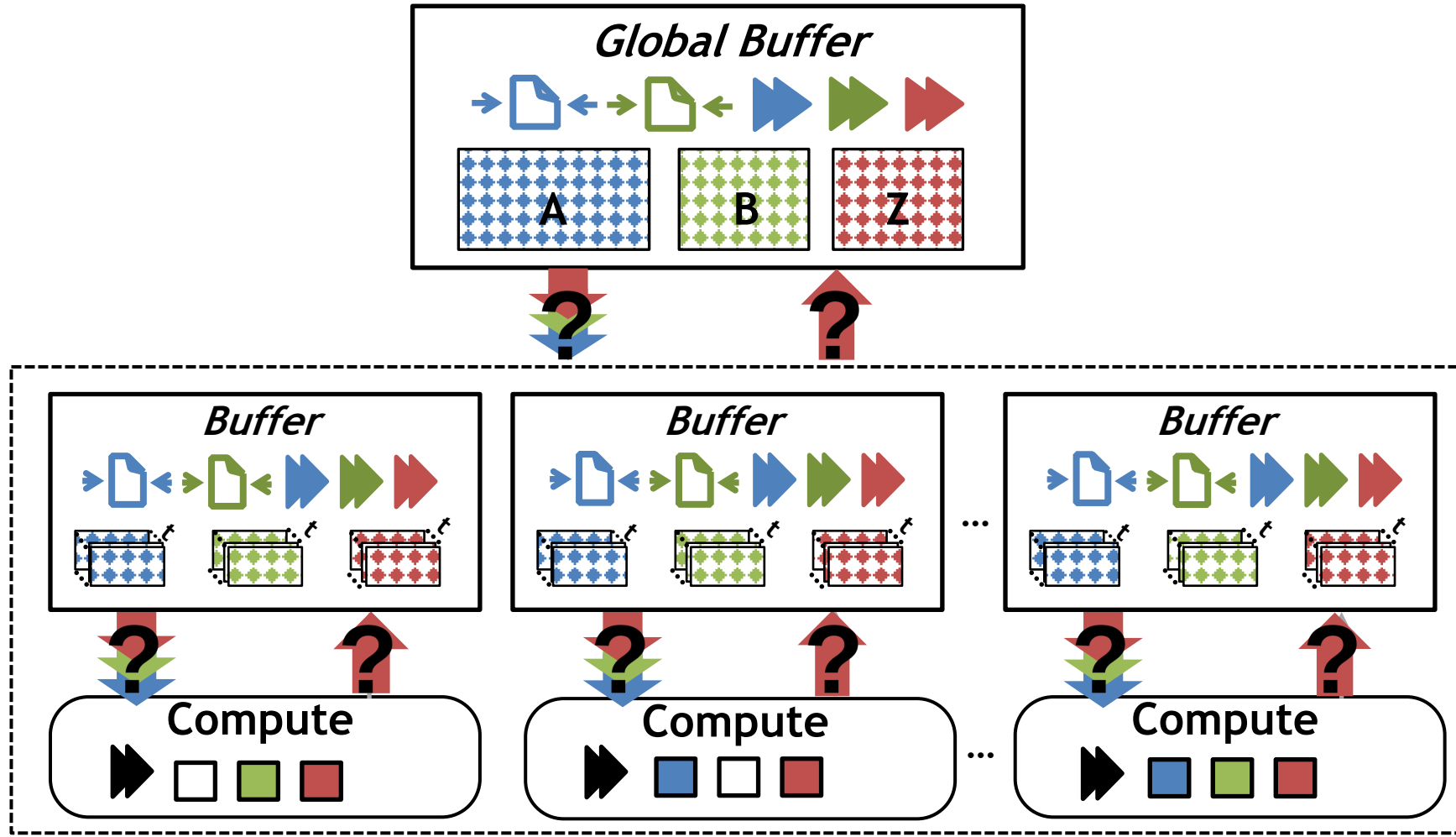
SAF Modeling Considers Additional Sparsity-Related Inputs



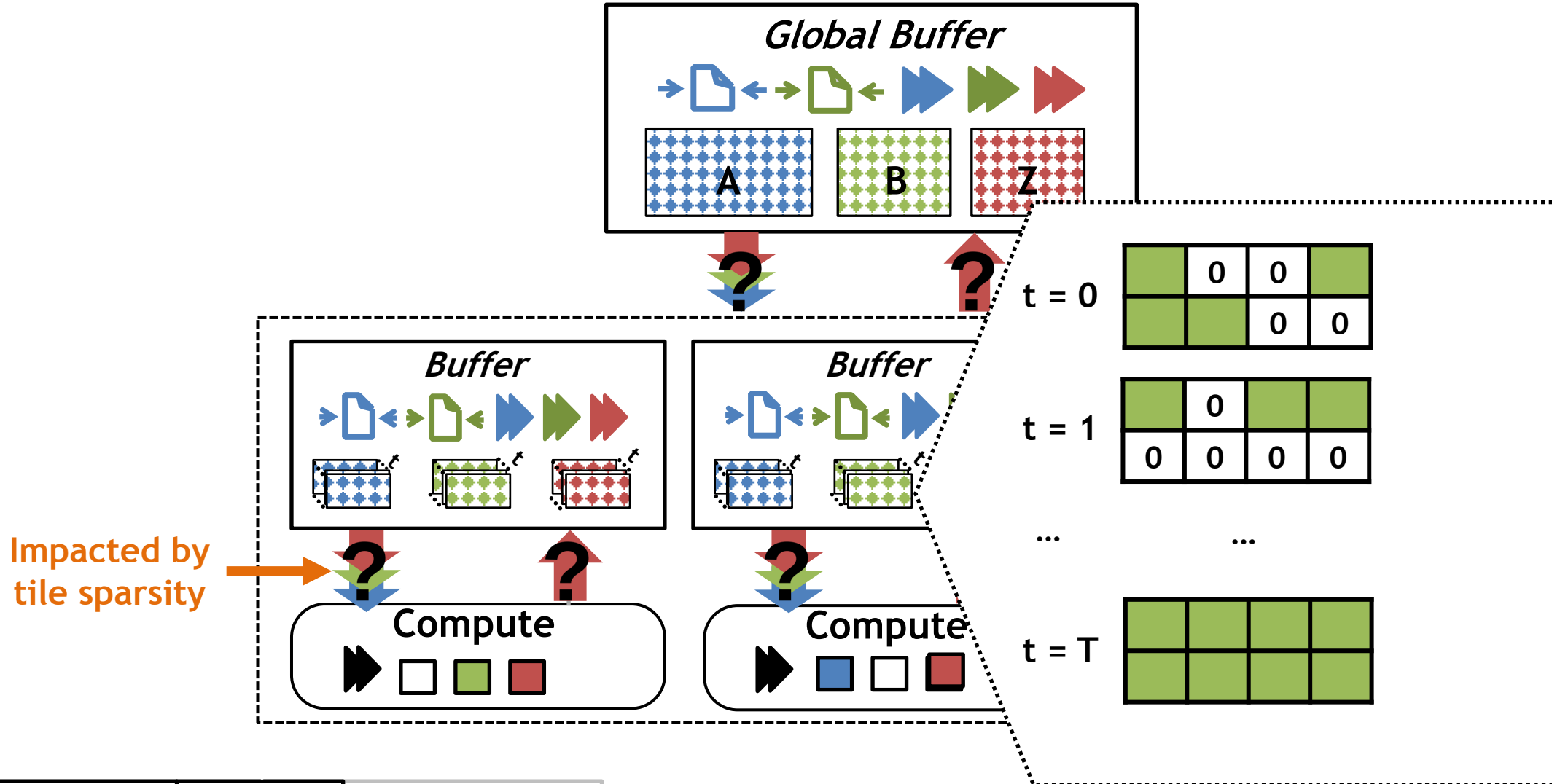
Sparse Tensors are also Tiled in a Similar Fashion



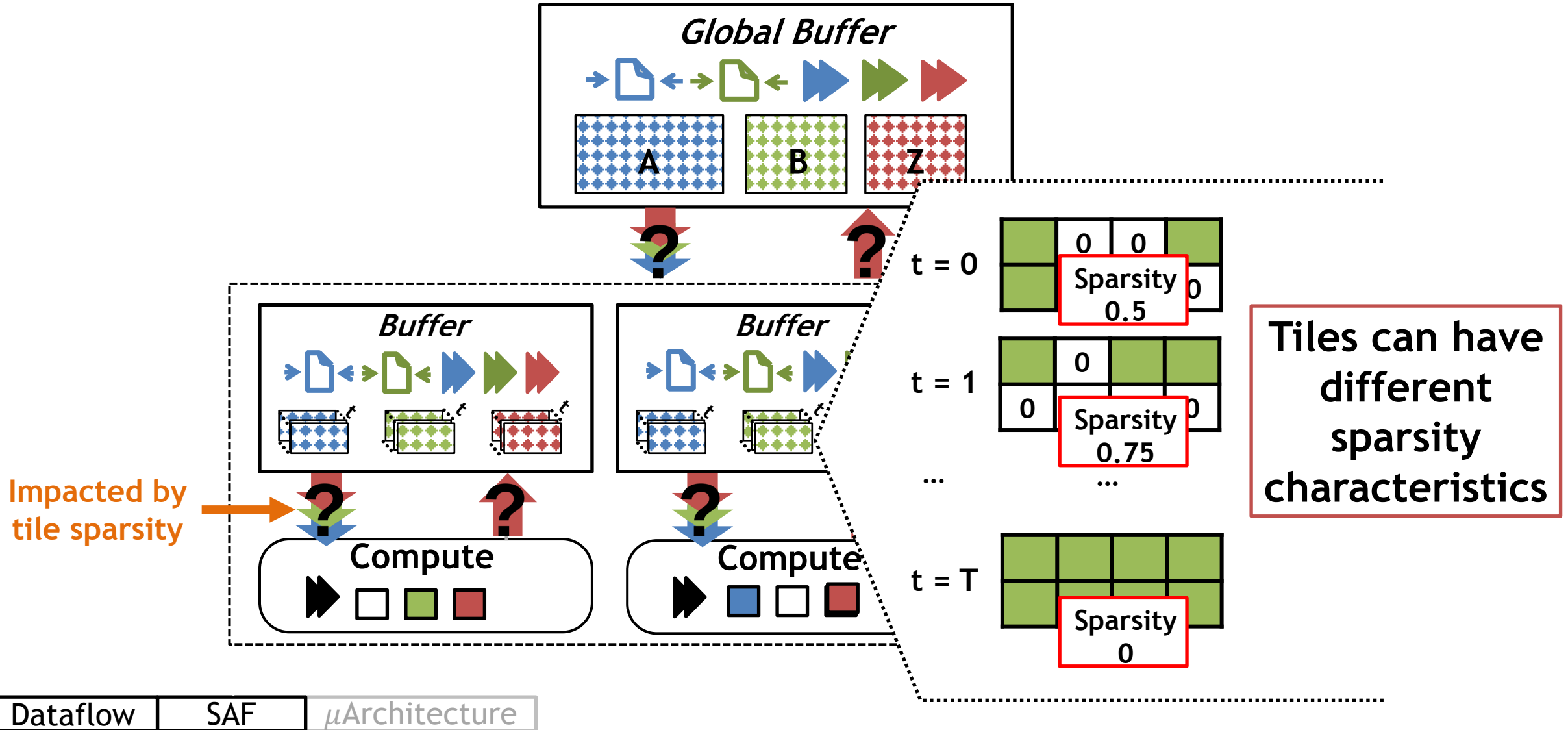
SAF Impacts the Original Dense Traffic



SAF Impact is Dependent on Each Tile's Sparsity

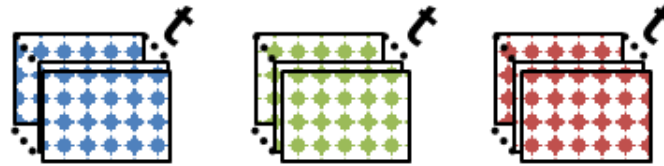


SAF Impact is Dependent on Each Tile's Sparsity



SAF Impact is Dependent on Each Tile's Sparsity

Naive Solution: Traverse Exact Data in Each Tile



Slow

Tiles can have different sparsity characteristics

Impacted by tile sparsity

Compute



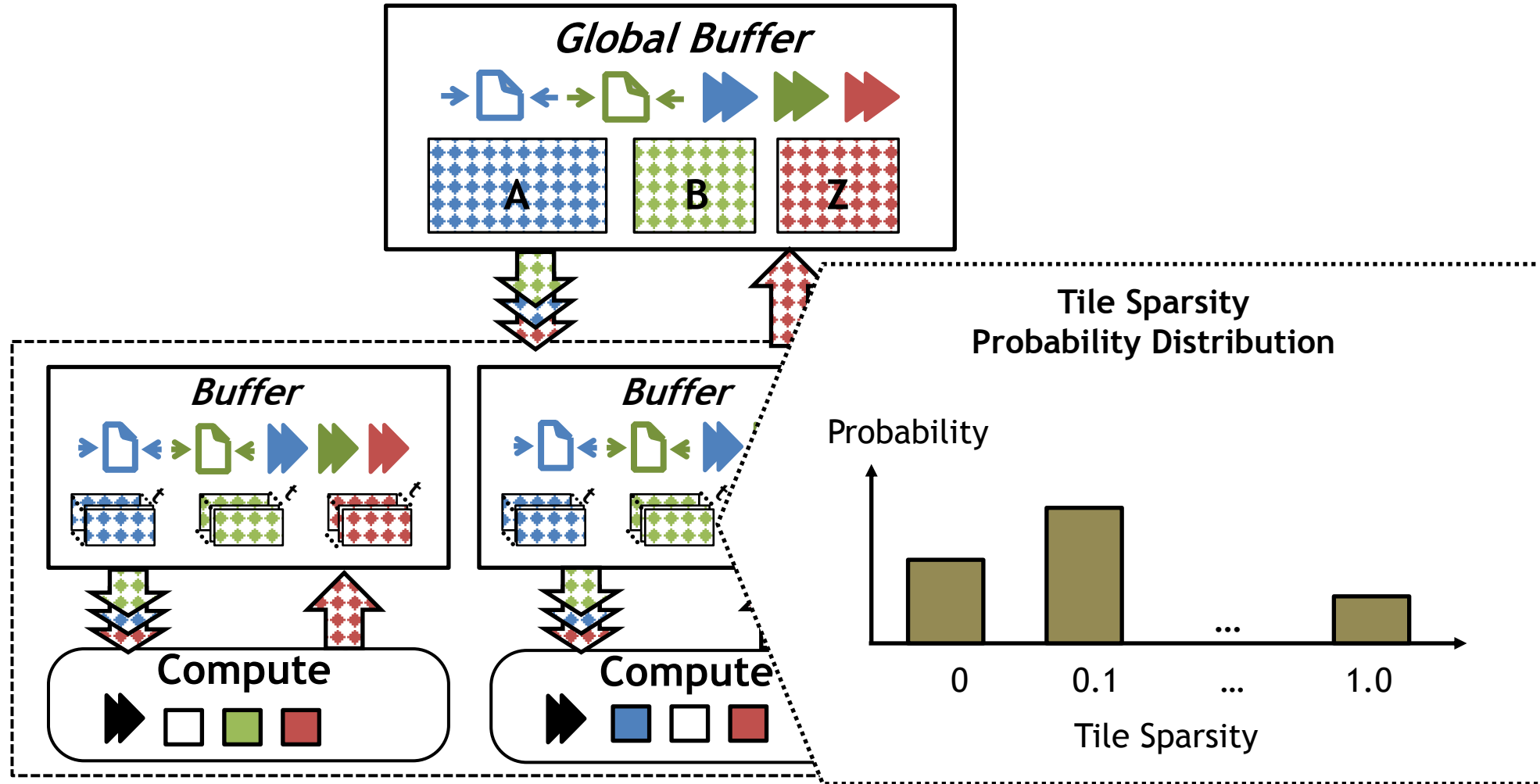
Compute



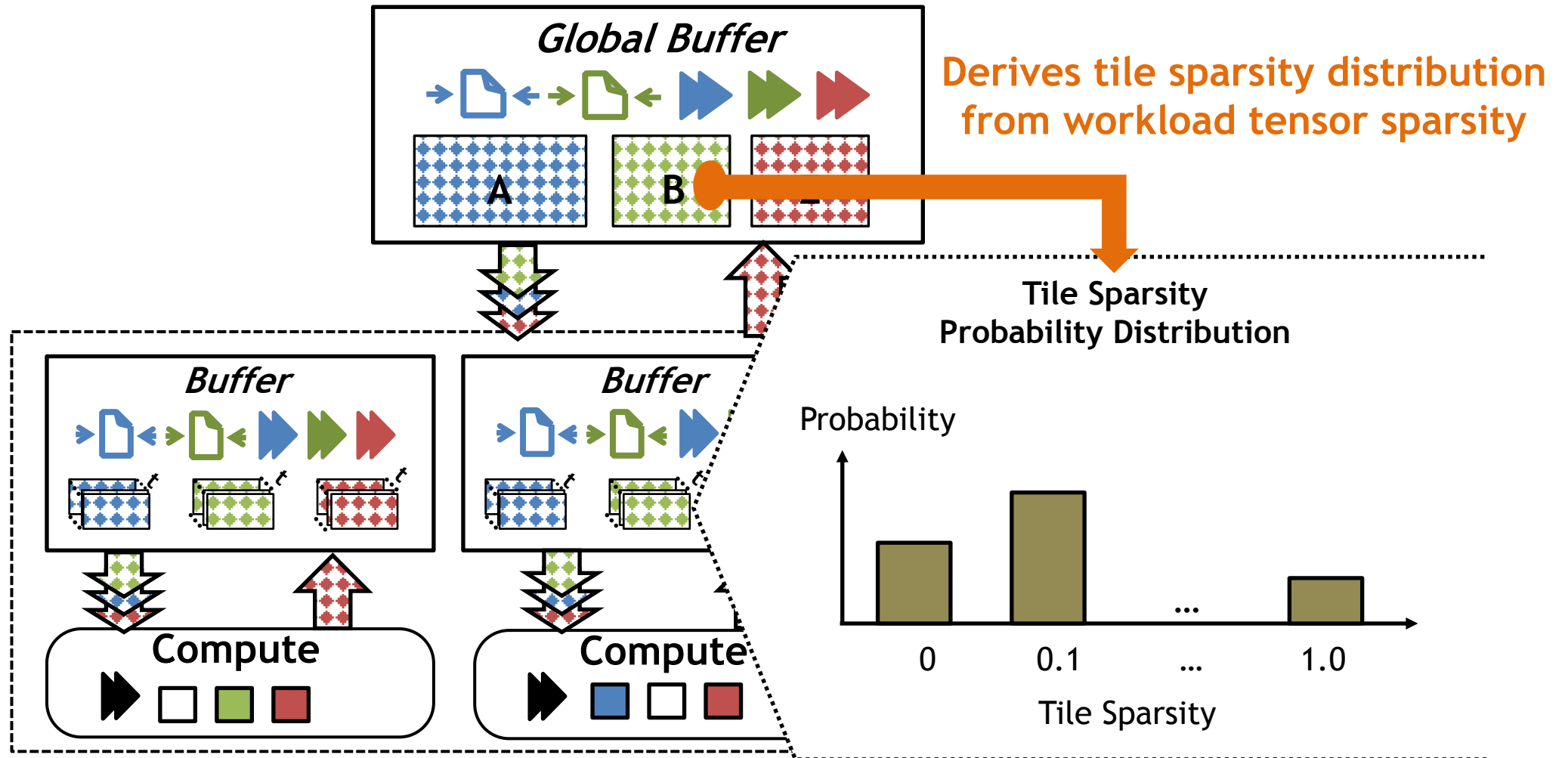
$t = T$

Sparsity
0

Employing Statistical Tile Sparsity Characterizations

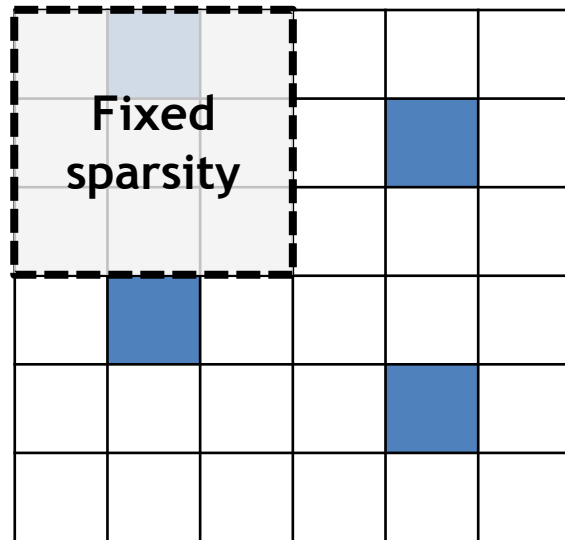


Employing Statistical Tile Sparsity Characterizations



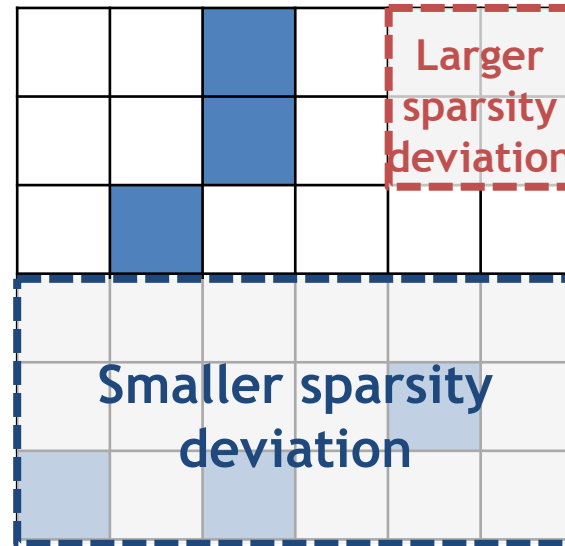
Supported Workload Tensor Sparsity Models

Fixed-Structured



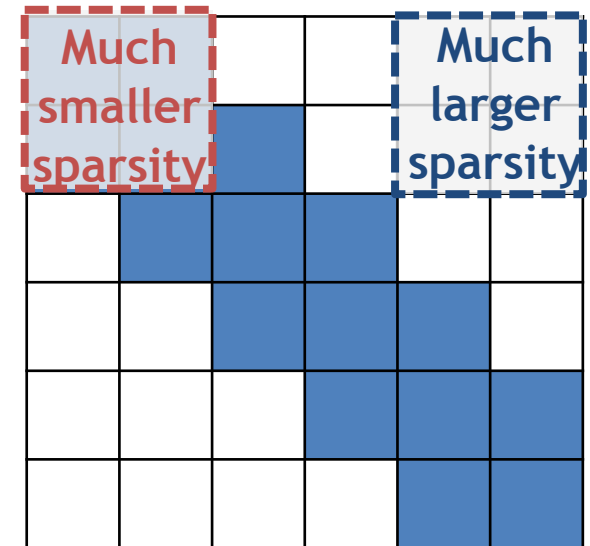
Structured Pruned DNNs

Uniform Random



Unstructured Pruned DNNs

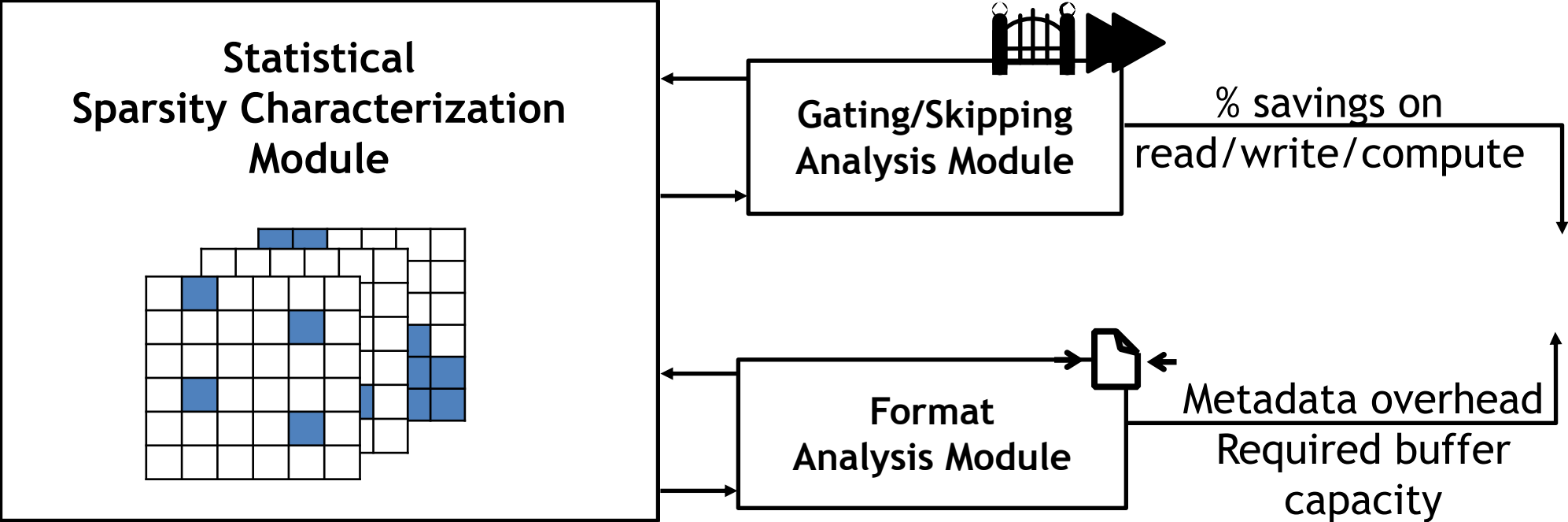
Banded Distribution



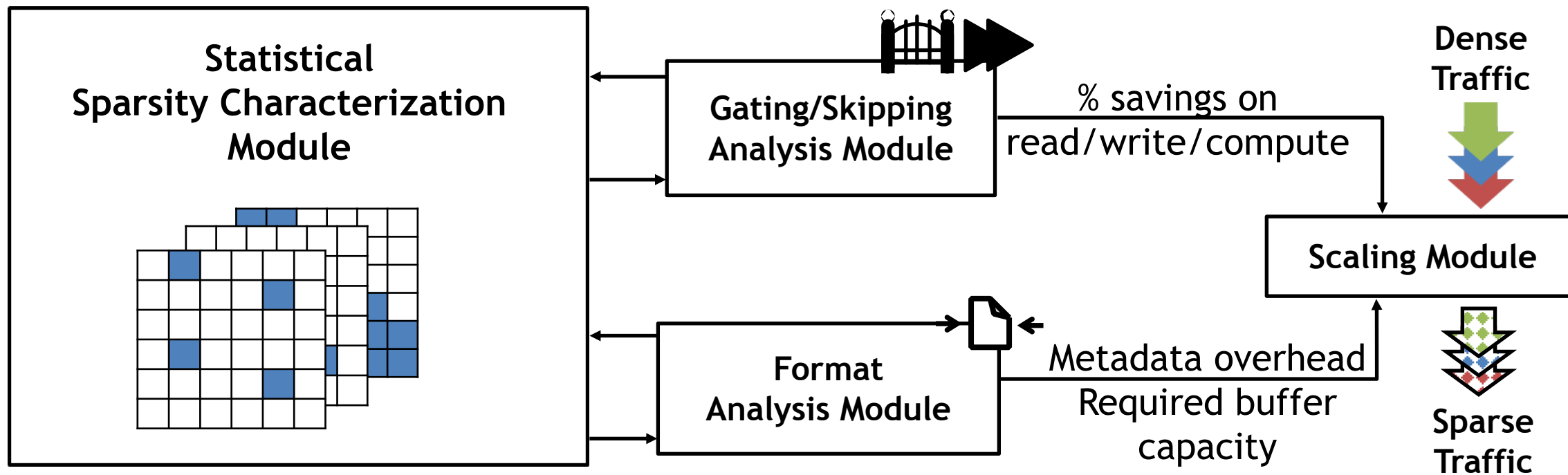
Circuit Simulations

Statistical modeling ensures both speed and accuracy

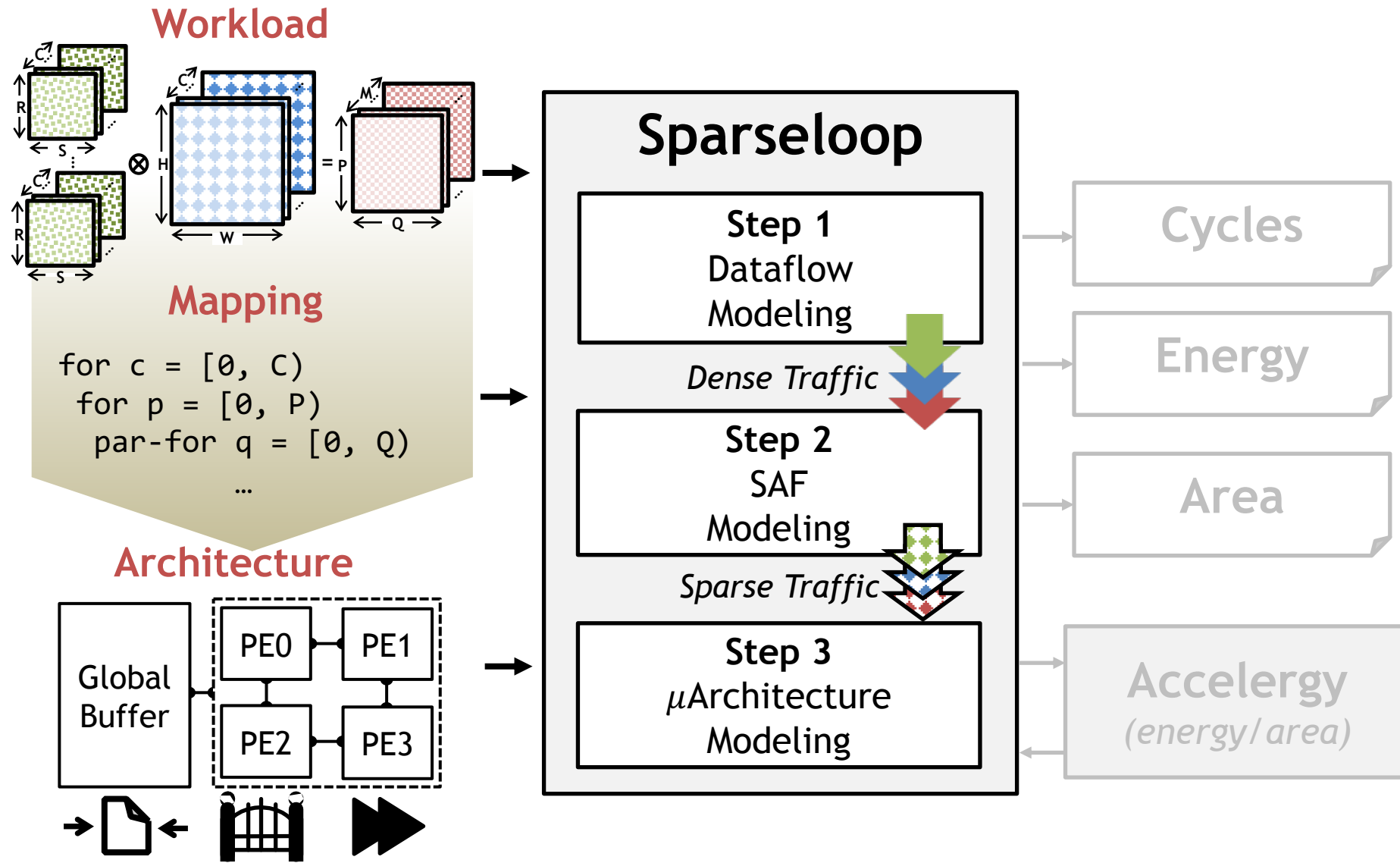
Per-Tile SAF Impact Analysis



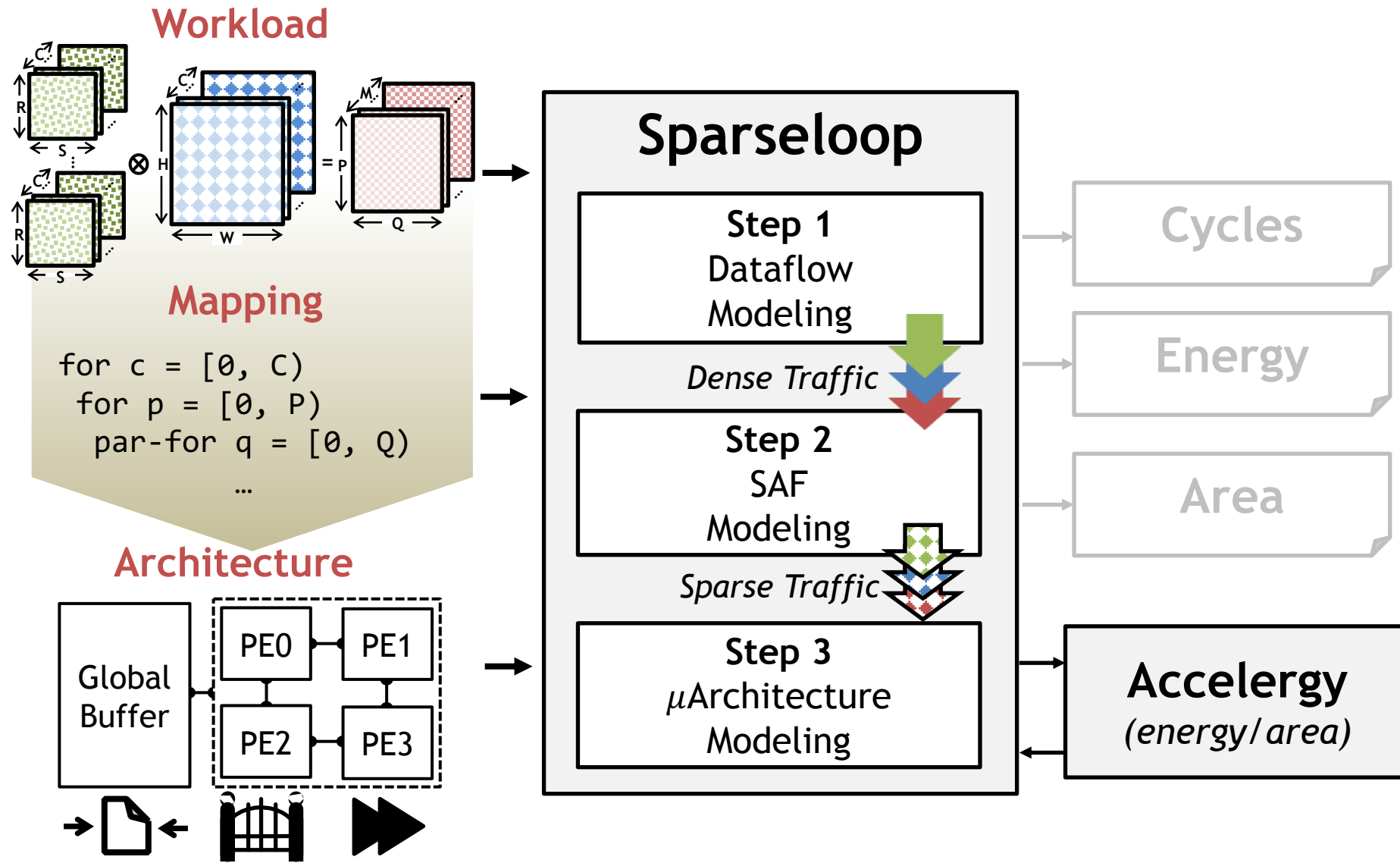
Filtering Dense Traffic to Produce Sparse Traffic



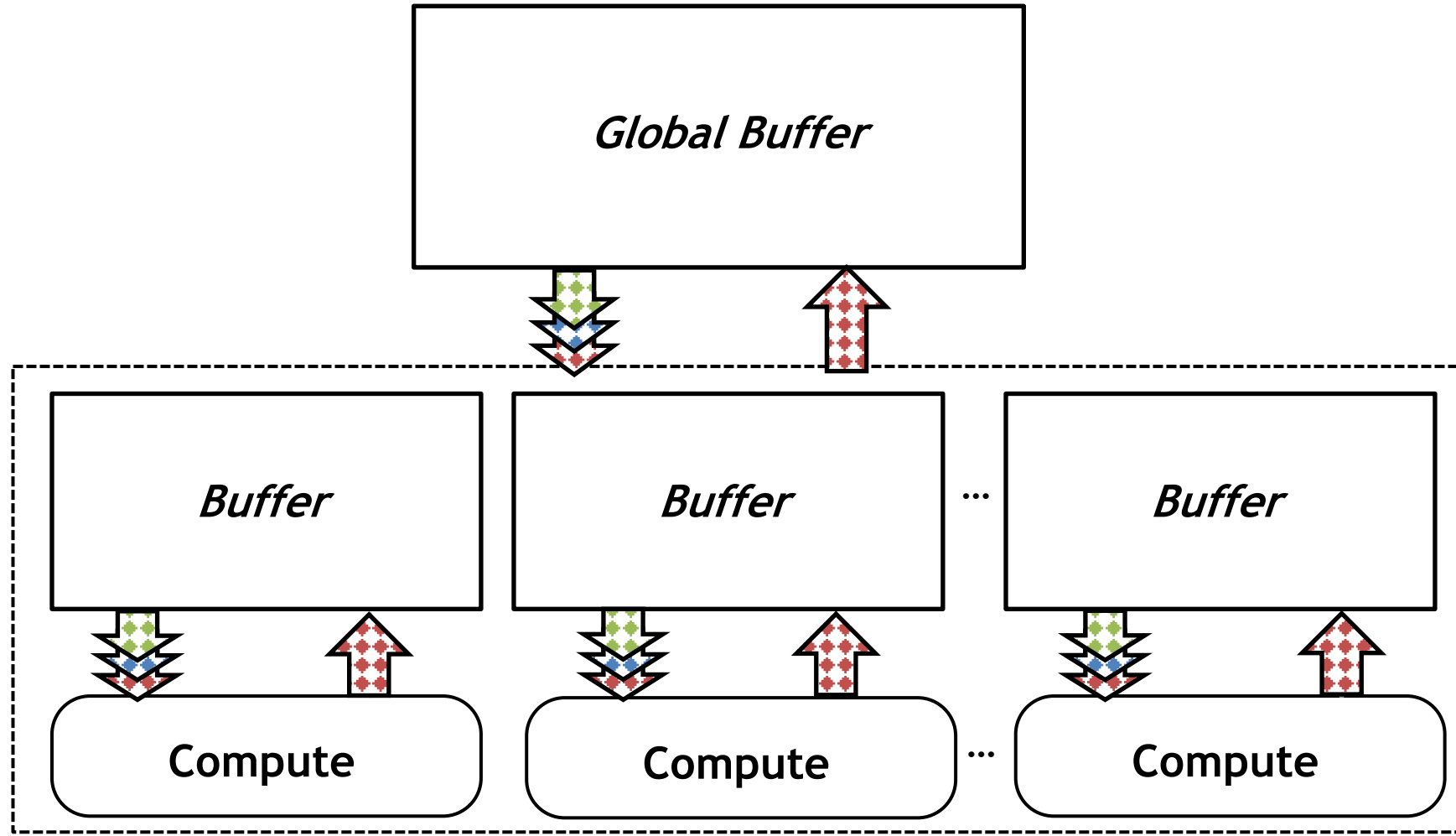
μ Architecture Modeling Considers Hardware Details



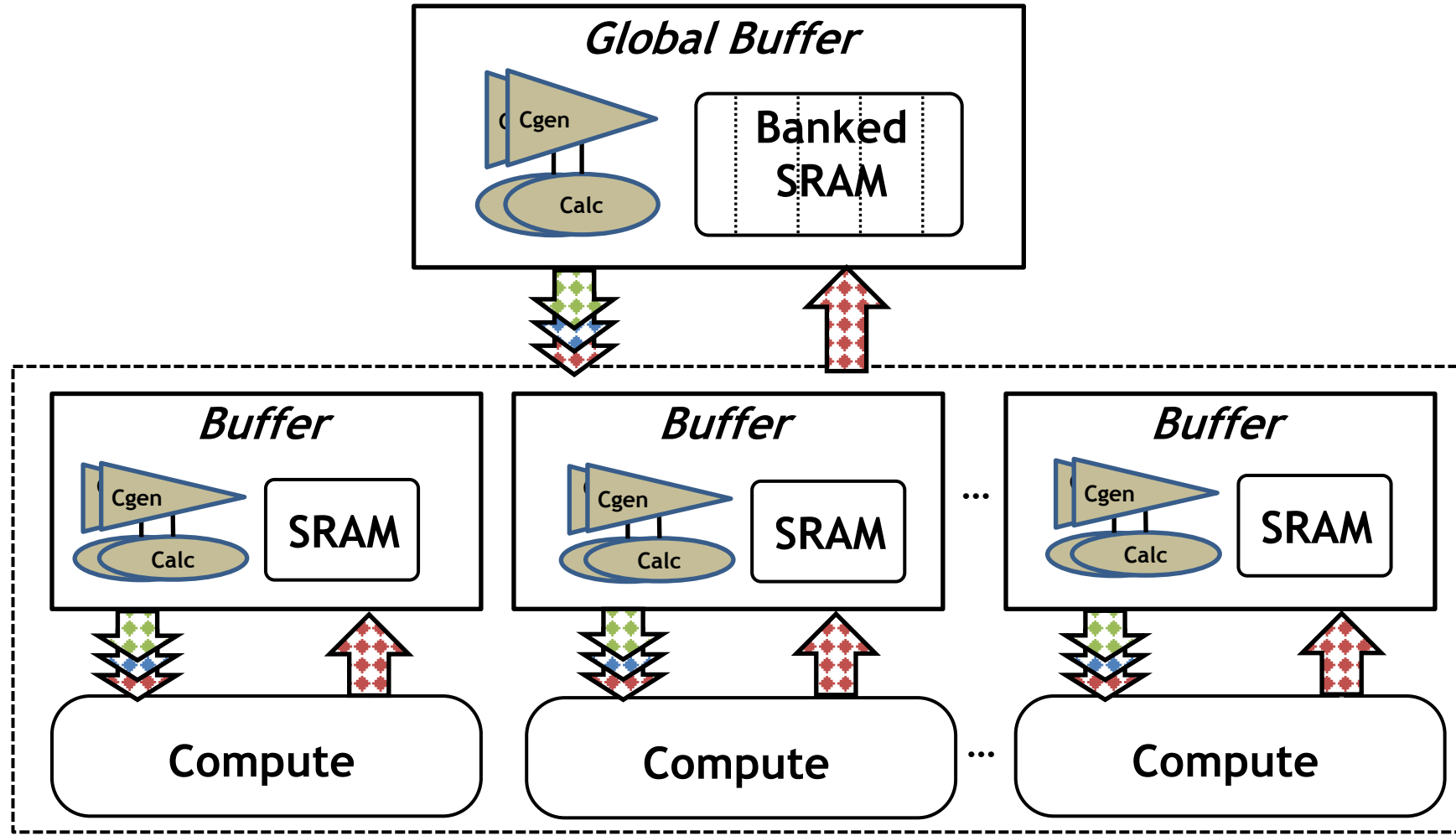
μ Architecture Modeling Considers Hardware Details



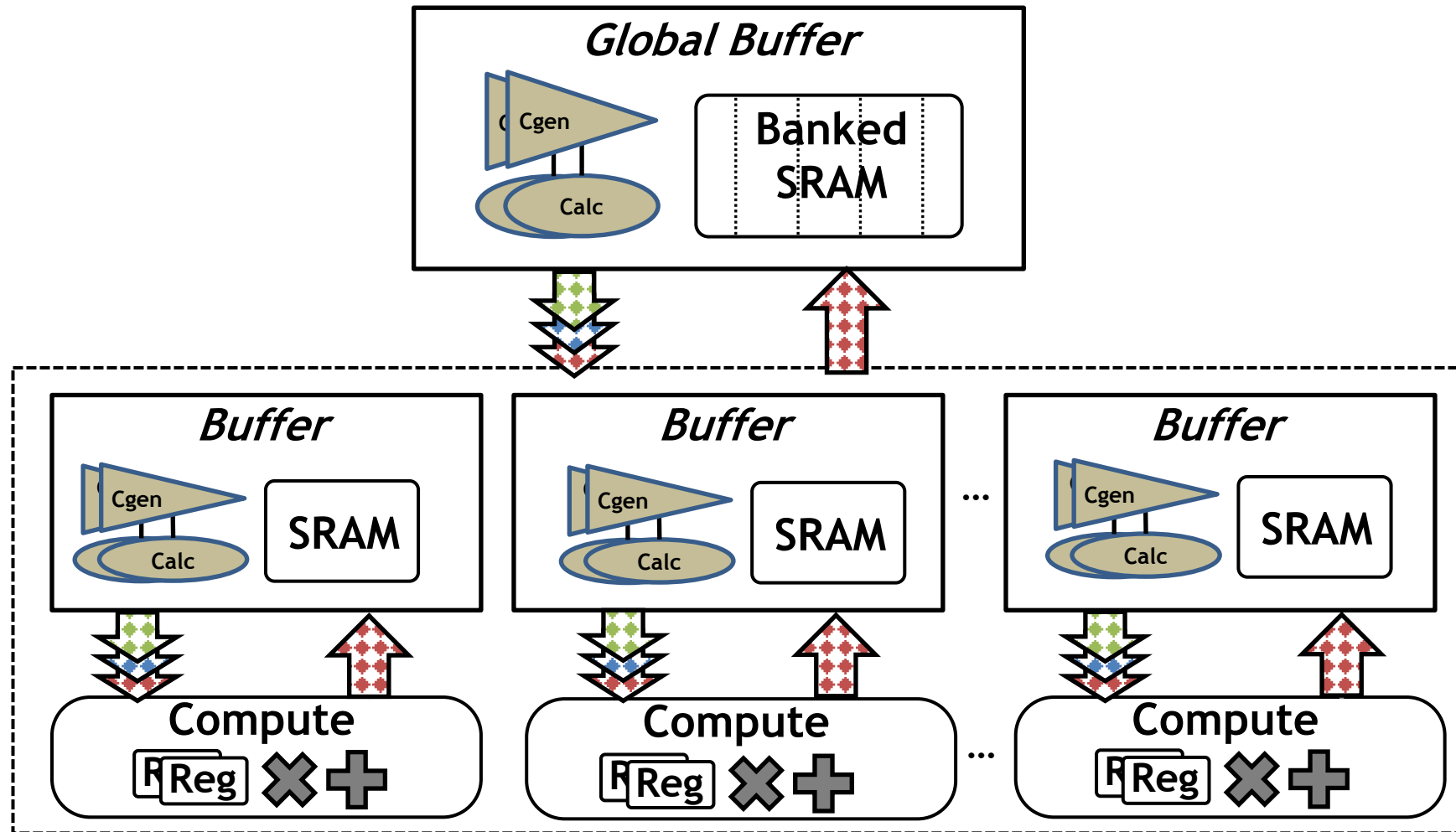
μ Architecture Modeling Considers Hardware Details



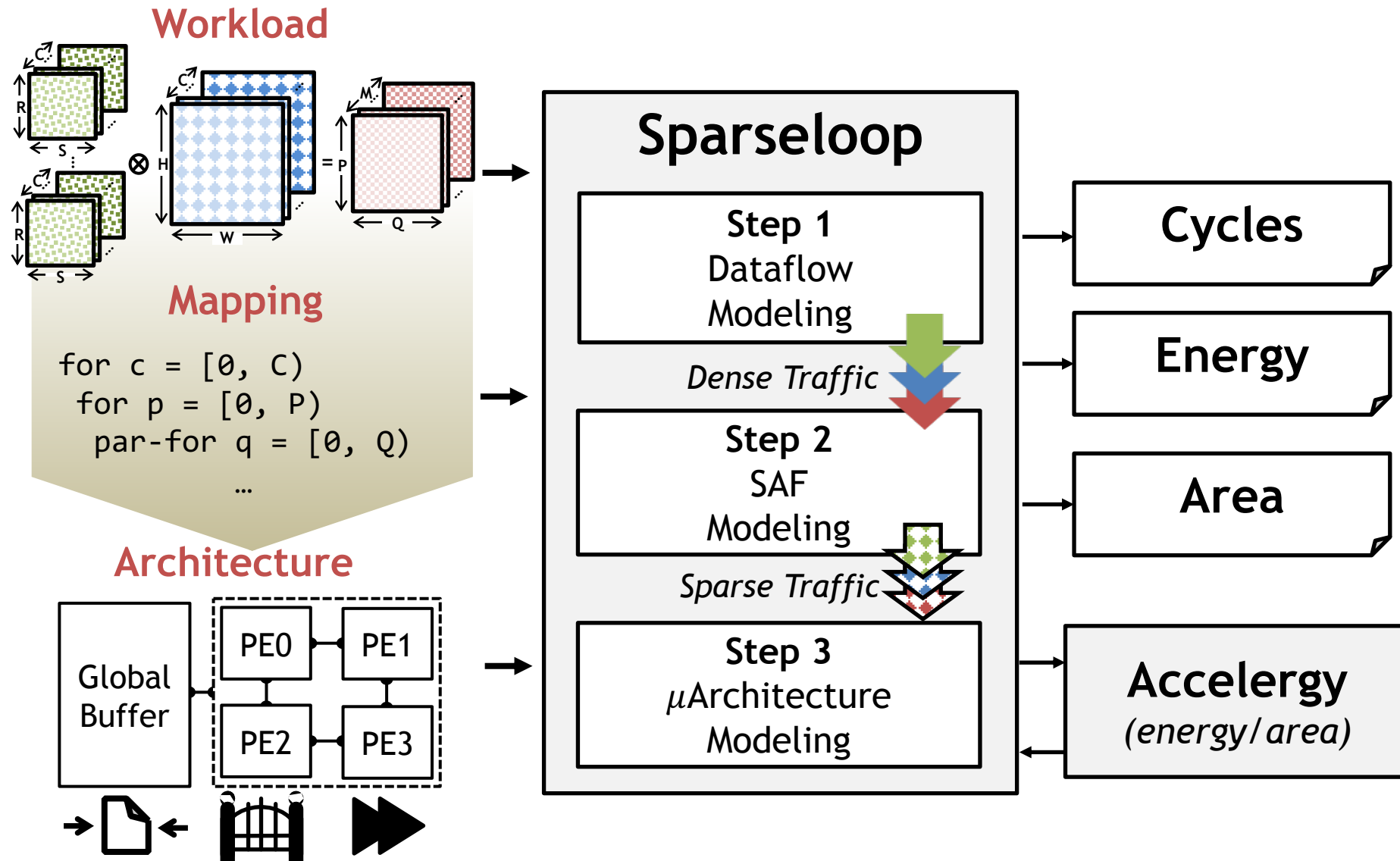
μ Architecture Modeling Considers Hardware Details



μ Architecture Modeling Considers Hardware Details



μ Architecture Modeling Drives Final Performance

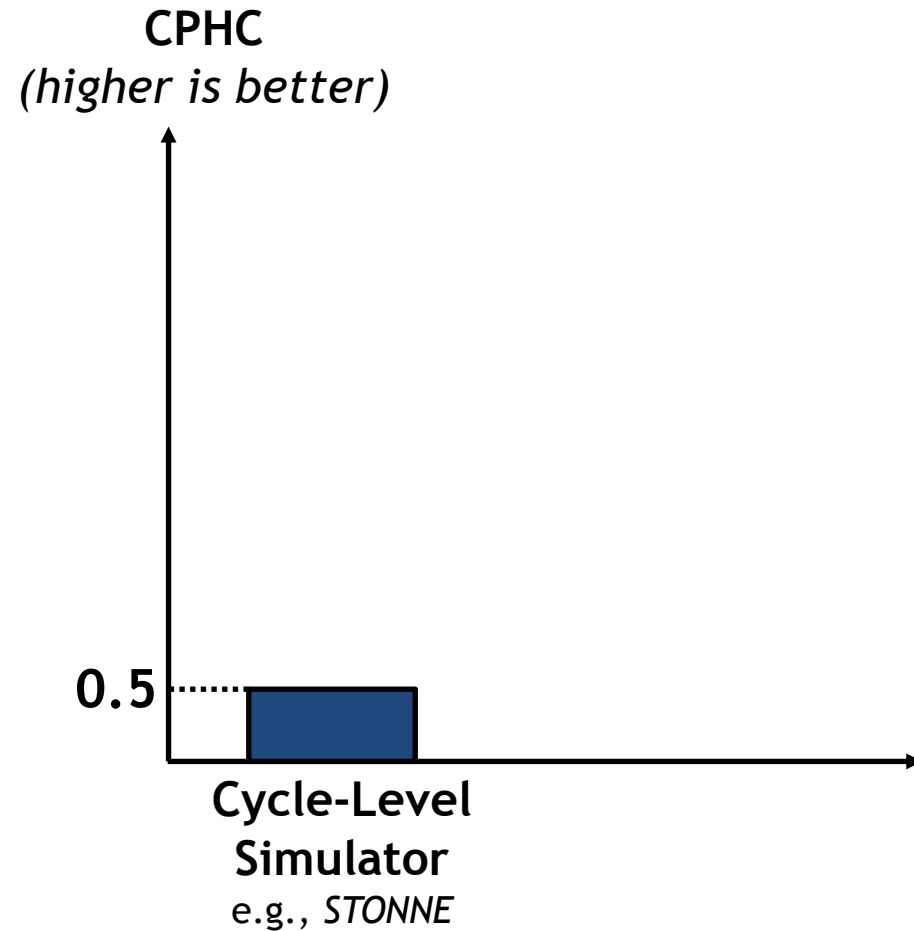


Experimental Results

Fast Simulation Speed

Evaluation metric: Computes Simulated Per **Host** Cycle (CPHC)

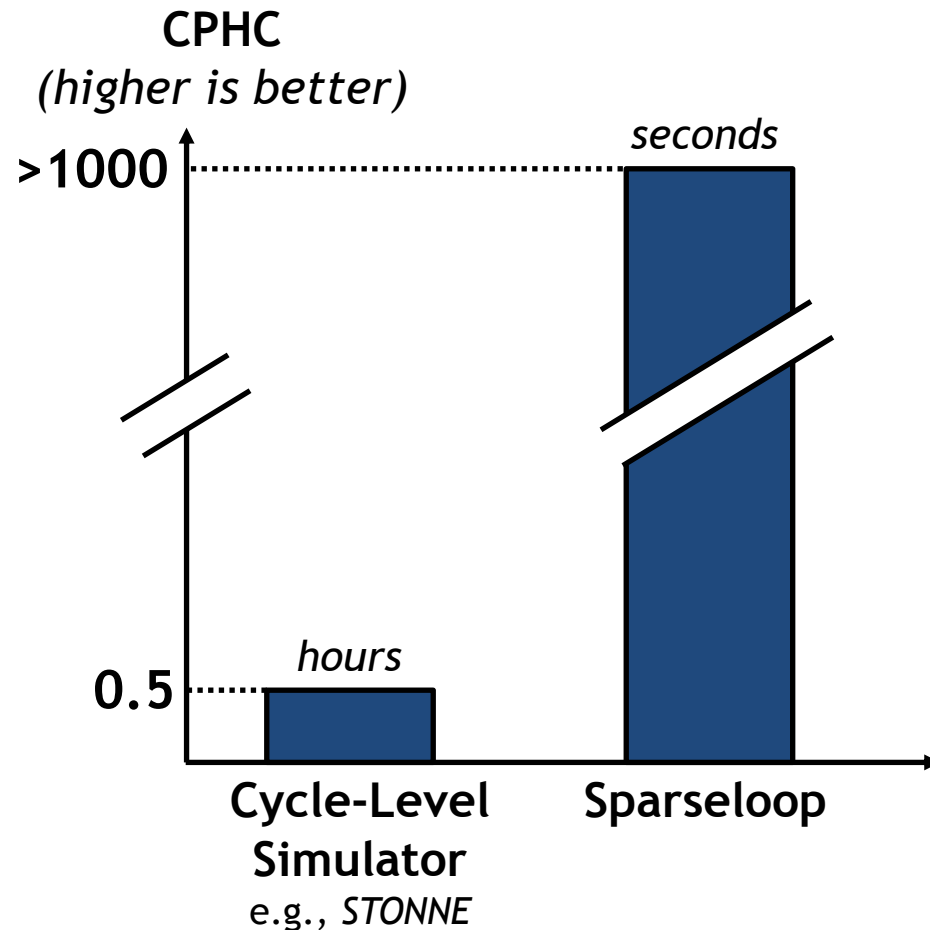
The machine that runs modeling framework, e.g., CPU



Fast Simulation Speed

Evaluation metric: Computes Simulated Per **Host** Cycle (CPHC)

The machine that runs modeling framework, e.g., CPU

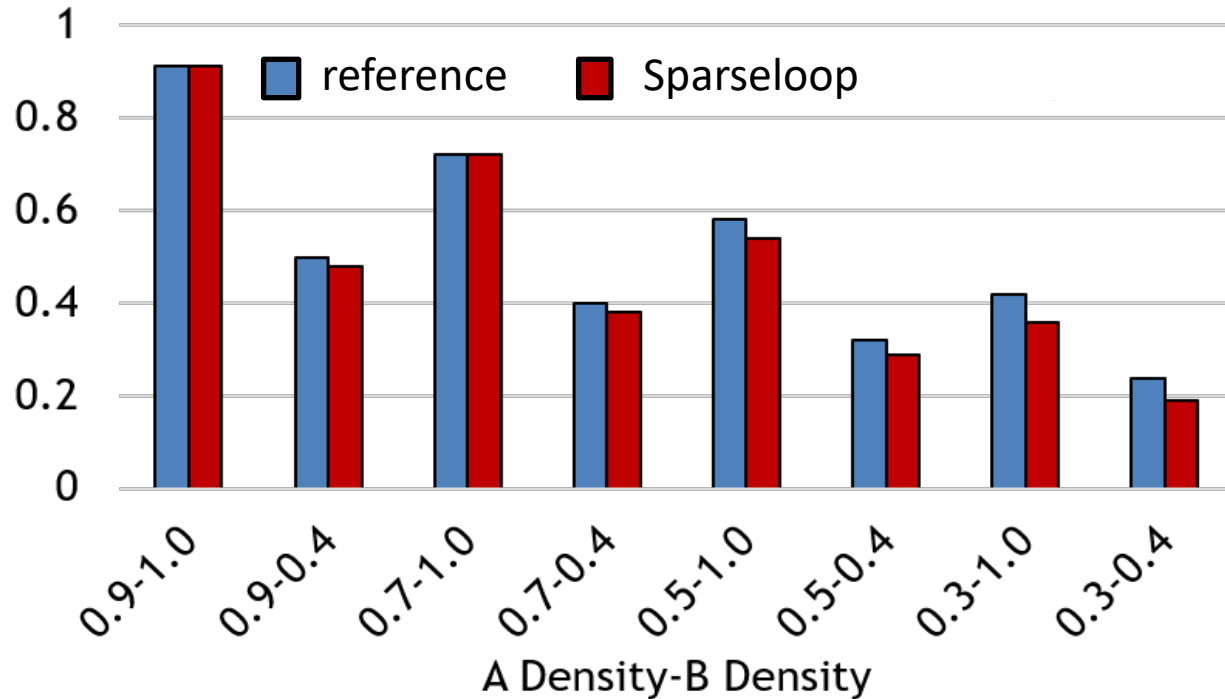


Across various sparse DNN accelerators, Sparseloop is more than 2000x faster than cycle-level simulation

Accurate Modeling

Example DSTC [ISCA21] Validation

Cycle Counts



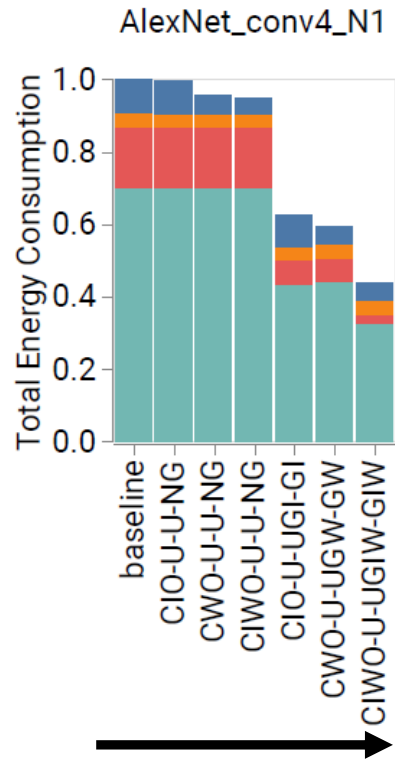
- Maintains absolute values
- Maintains relative trends

Maintains <8% average error in cycle counts and energy consumption across representative DNN accelerators

Rapid Exploration of Various Designs

Components

■ DRAM ■ GLB ■ MACs ■ spads



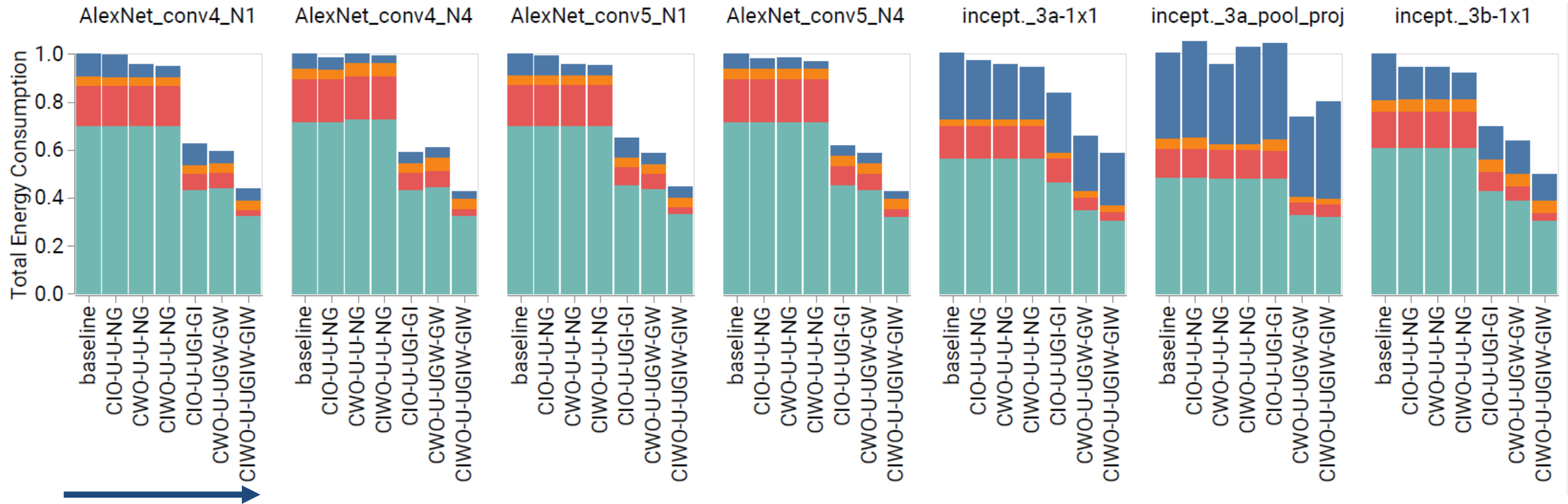
Various combinations of SAFs

Rapid Exploration of Various Designs

Various workloads

Components

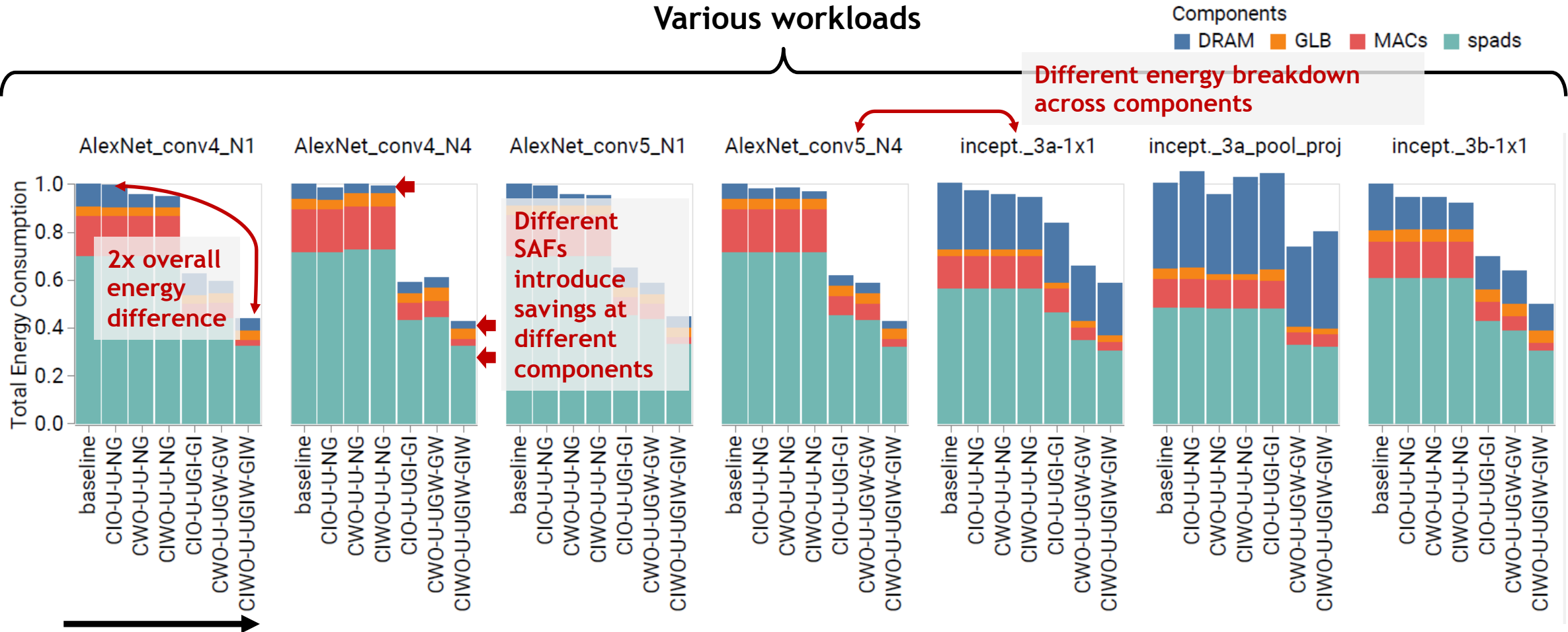
■ DRAM
 ■ GLB
 ■ MACs
 ■ spads



Various combinations of SAFs

Rapid Exploration of Various Designs

Various workloads



Various combinations of SAFs

Sparseloop helps to quickly identify the savings and overheads for the diverse implementations

Taxonomy/Sparseloop Summary

Systematic Design Space Descriptions

- 3 sparse acceleration features (SAFs)
- Systematic description by assigning SAFs to each architecture level

Sparseloop Modeling Framework

- Progressive modeling with tractable complexity
- Statistical modeling with balanced accuracy and speed
- Ensures accuracy, speed, and flexibility

Taxonomy/Sparseloop Summary

Systematic Design Space Descriptions

- 3 sparse acceleration features (SAFs)
- Systematic description by assigning SAFs to each architecture level

Sparseloop Modeling Framework

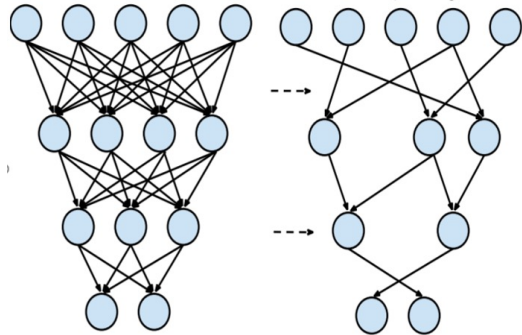
- >2000x faster compared to cycle-level simulations
- <8% average error across various designs/workloads
- Demonstrated flexible design space exploration

III. Efficient and Flexible DNN Accelerator Design

- Hierarchical Structured Sparsity
- Modularized Acceleration Design Methodology
- Experimental Results

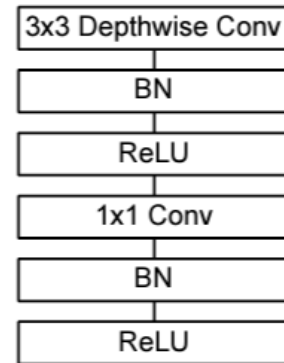
Different DNN Optimizations Introduce Different Sparsity

Optimizations to Reduce Model Size



Pruning
Techniques
[Han, NeurIPS15]

Introduces
Sparse Weights

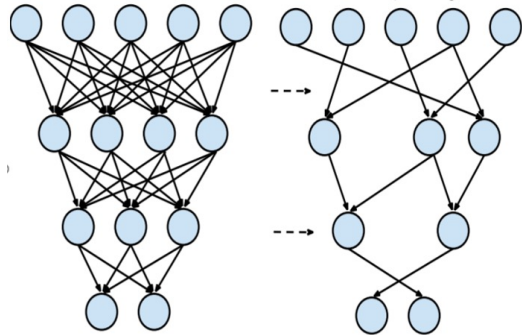


Depth-wise
Separable Layers
[Howard, CVPR17]

Introduces
Dense Weights

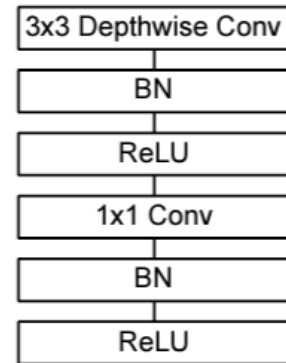
Different DNN Optimizations Introduce Different Sparsity

Optimizations to Reduce Model Size



Pruning
Techniques
[Han, NeurIPS15]

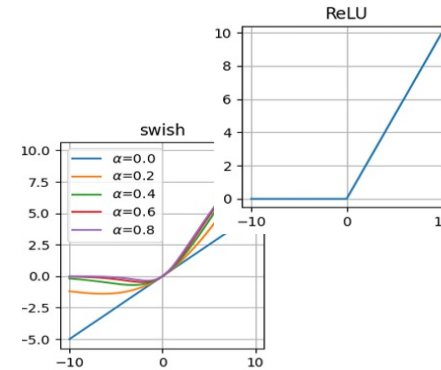
Introduces
Sparse Weights



Depth-wise
Separable Layers
[Howard, CVPR17]

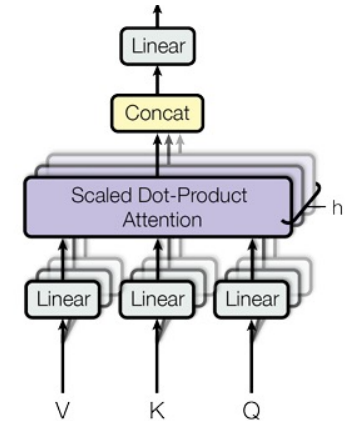
Introduces
Dense Weights

Optimizations to Improve Accuracy



Activation
Functions
[Apicella, NN21]

Introduces
Dense/Sparse Activations

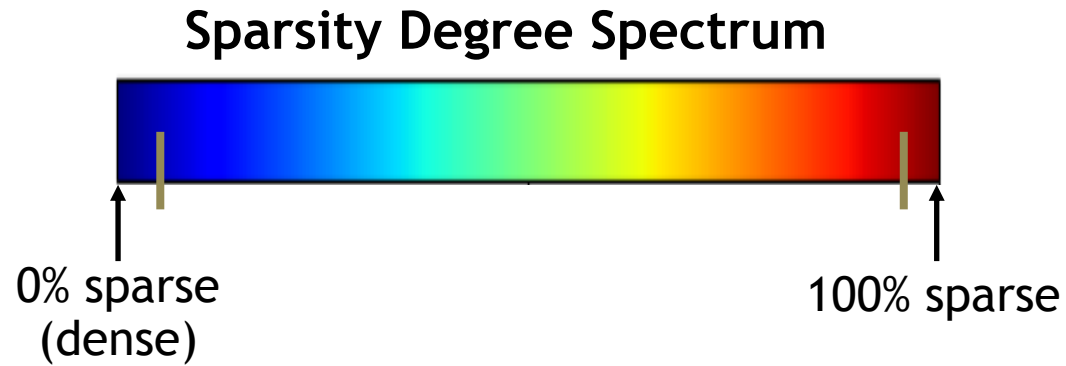


Attention-based
Modules
[Vaswani, NeurIPS17]

Introduces
Dense Act./Weights

Modern DNNs can weights and activations that are either dense or sparse with various sparsity degrees

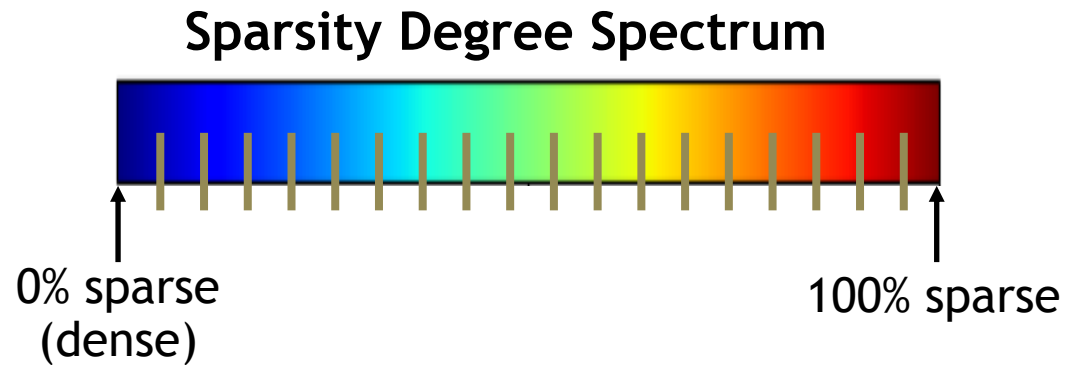
Requirements for an Ideal Sparse DNN Accelerator



Flexible

*exploit a wide range of
many sparsity degrees*

Requirements for an Ideal Sparse DNN Accelerator



Flexible

exploit a wide range of many sparsity degrees

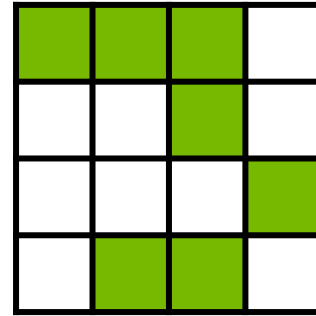
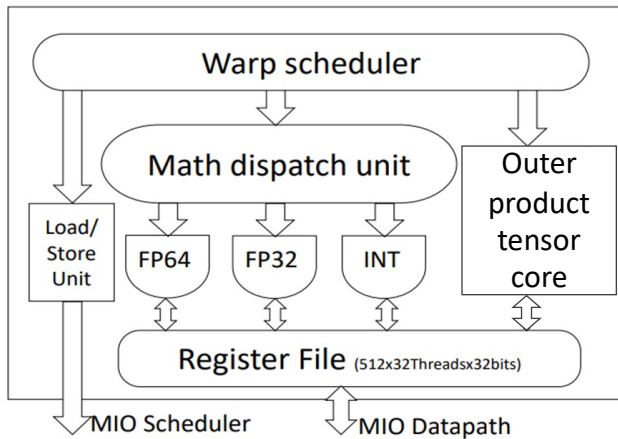


Efficient

low hardware overhead SAF implementations

Existing Works Do Not Meet Such Requirements

Unstructured Sparse Accelerators



Unstructured sparse

Dual-Side Sparse Tensor Core (DSTC) [Wang, ISCA21]

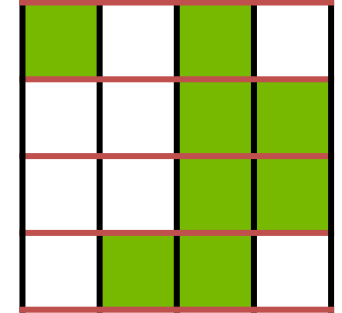
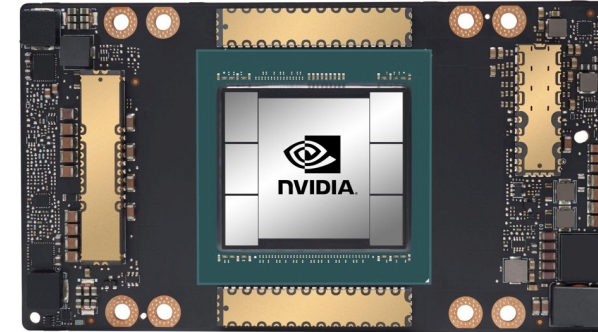
Sparsity Degree Spectrum



Continuously Translated into Savings

- Inefficient
- + Flexible

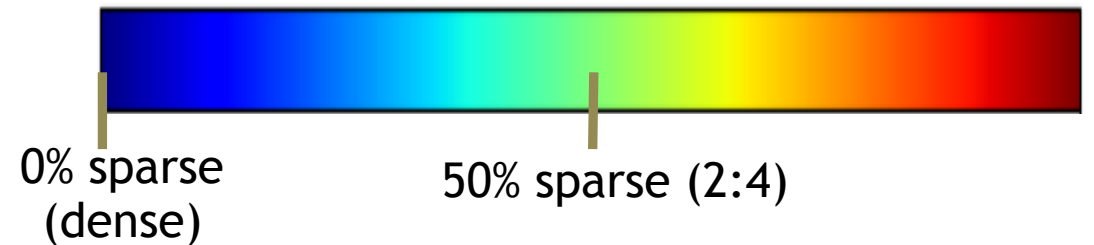
Structured Sparse Accelerators



Per-row
2:4 structured sparse
(G:H pattern)

NVIDIA Sparse Tensor Core (STC) [NVIDIA, TechReport20]

Sparsity Degree Spectrum



0% sparse
(dense)

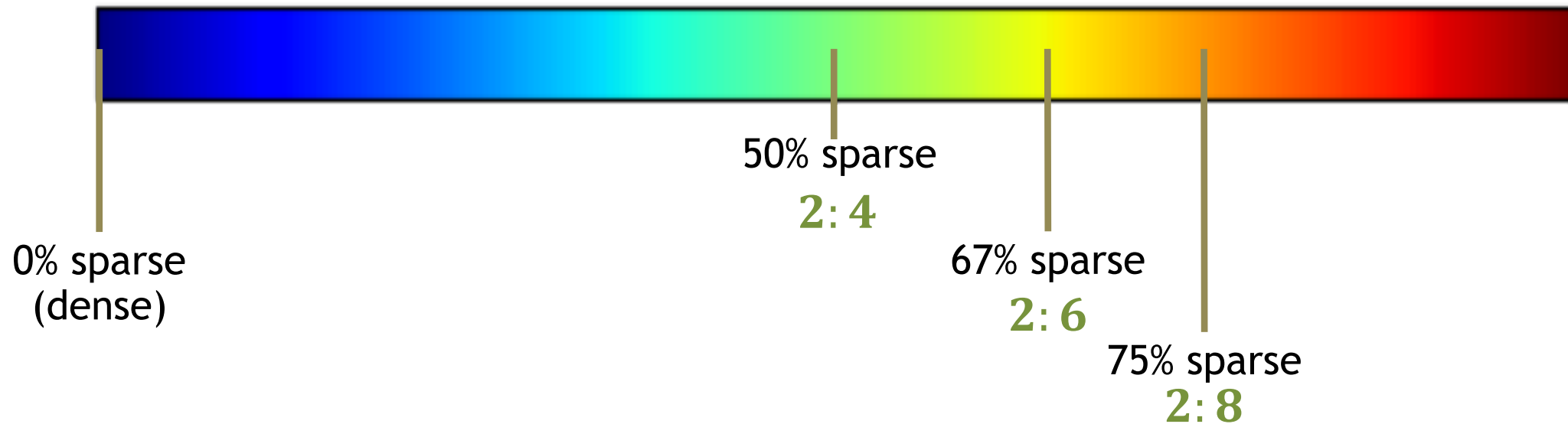
50% sparse (2:4)

- + Efficient
- Inflexible

Naïve Way to Increase Flexibility Structured Sparse Designs

Extend the Number of G:H Ratios Supported

Sparsity Degree Spectrum



Not Scalable

Hardware complexity increases approximately in proportion to the number of sparsity degrees



Our Proposal

Efficient and Flexible DNN Acceleration with Hierarchical Structured Sparsity



Hierarchical Structured Sparsity (HSS)

Compose G:H sparsity patterns in a hierarchical fashion

$$\text{N-Rank HSS: } \underset{\text{Rank } N-1}{\text{G:H}} \rightarrow \underset{\text{Rank } N-2}{\text{G:H}} \dots \rightarrow \underset{\text{Rank } 0}{\text{G:H}}$$

What does a $3:4 \rightarrow 2:4$ pattern look like?



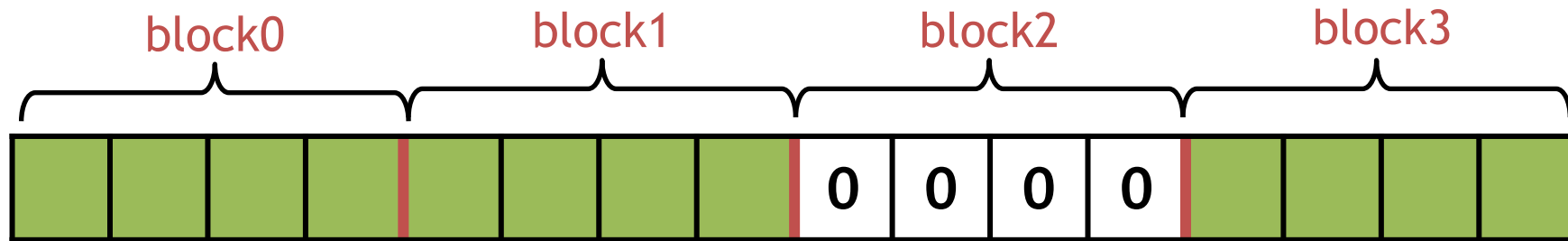
Dense Vector

Hierarchical Structured Sparsity (HSS)

Compose G:H sparsity patterns in a hierarchical fashion

*What does a **3:4** → 2:4 pattern look like?*

Rank1: 3 nonempty blocks out of the 4 blocks



Vector with Rank1 Sparsity Applied

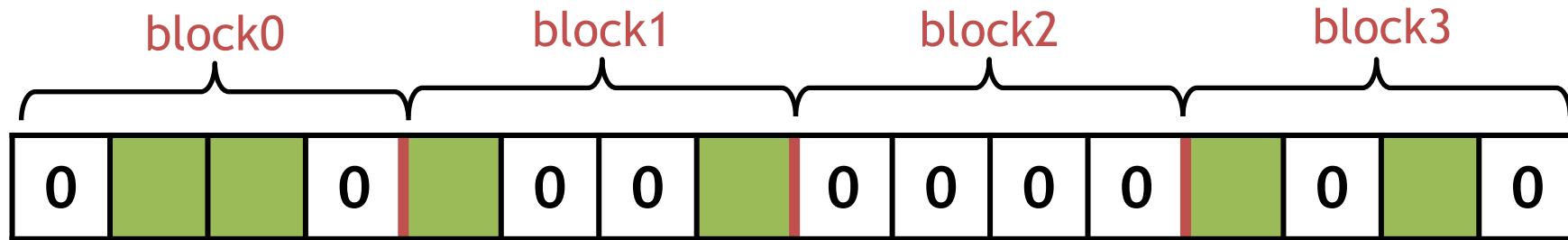
Hierarchical Structured Sparsity (HSS)

Compose G:H sparsity patterns in a hierarchical fashion

What does a 3:4 → 2:4 pattern look like?

Rank1: 3 nonempty blocks out of the 4 blocks

Rank0: 2 nonzero values out of 4 values within the block

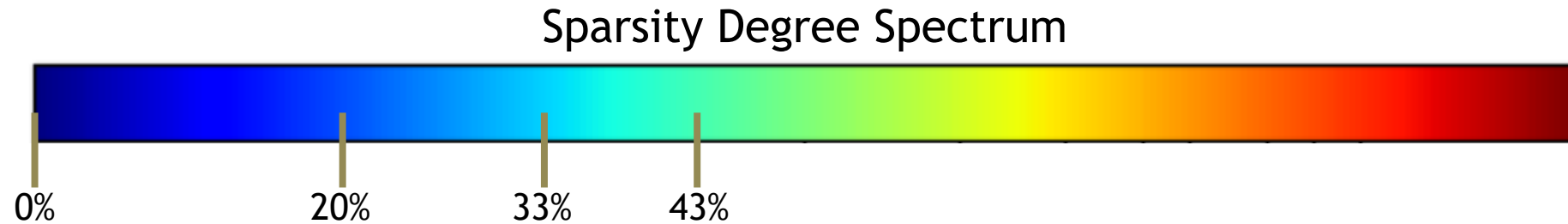


Vector with Both Ranks' Sparsity Applied

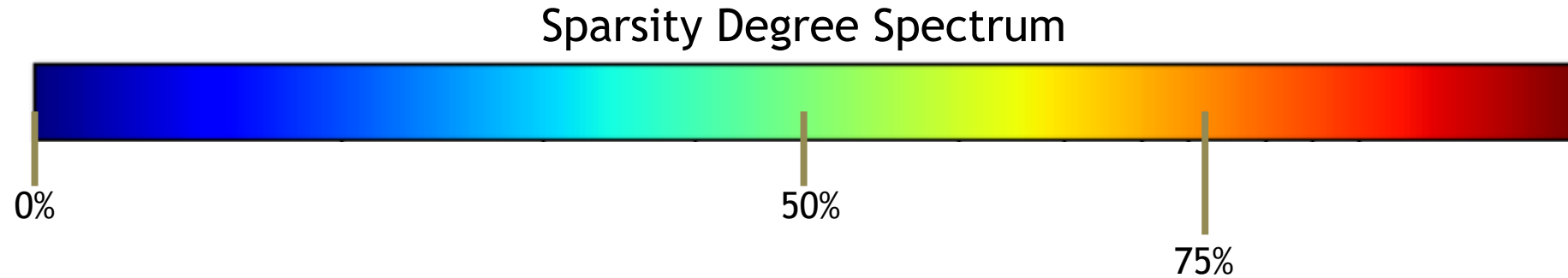
HSS Introduces A Flexible Way to Express Sparsity Degrees

4
sparsity
degrees

| Rank 1 | | | |
|--------|-------|-------|-------|
| 4:4 | 4:5 | 4:6 | 4:7 |
| (0%) | (20%) | (33%) | (43%) |



HSS Introduces A Flexible Way to Express Sparsity Degrees



HSS Introduces A Flexible Way to Express Sparsity Degrees



Multiplication of Fractions

4:5-2:4
(60%)

Sparsity Degree Spectrum



60%

HSS Introduces A Flexible Way to Express Sparsity Degrees



Multiplication of Fractions

4:5-2:4
(60%)

4:6-1:4
(83%)

Sparsity Degree Spectrum



HSS Introduces A Flexible Way to Express Sparsity Degrees

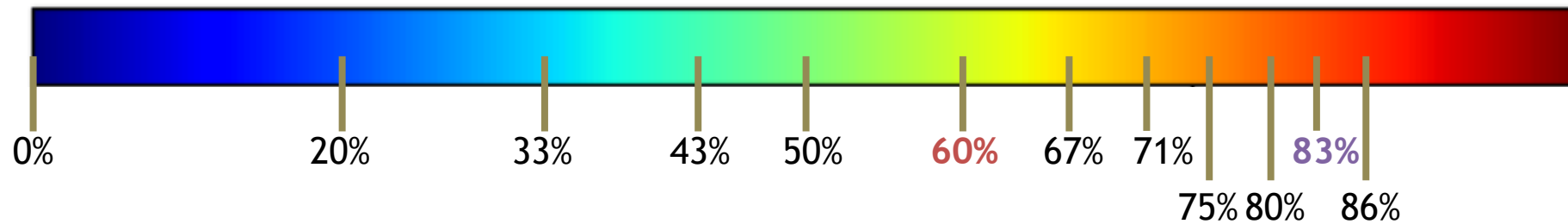


Multiplication of Fractions

12 sparsity degrees

| | | | | | | | | | | | |
|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| 4:4-4:4 | 4:5-4:4 | 4:6-4:4 | 4:7-4:4 | 4:4-2:4 | 4:5-2:4 | 4:6-2:4 | 4:7-2:4 | 4:4-1:4 | 4:5-1:4 | 4:6-1:4 | 4:7-1:4 |
| (0%) | (20%) | (33%) | (43%) | (50%) | (60%) | (67%) | (71%) | (75%) | (80%) | (83%) | (86%) |

Sparsity Degree Spectrum

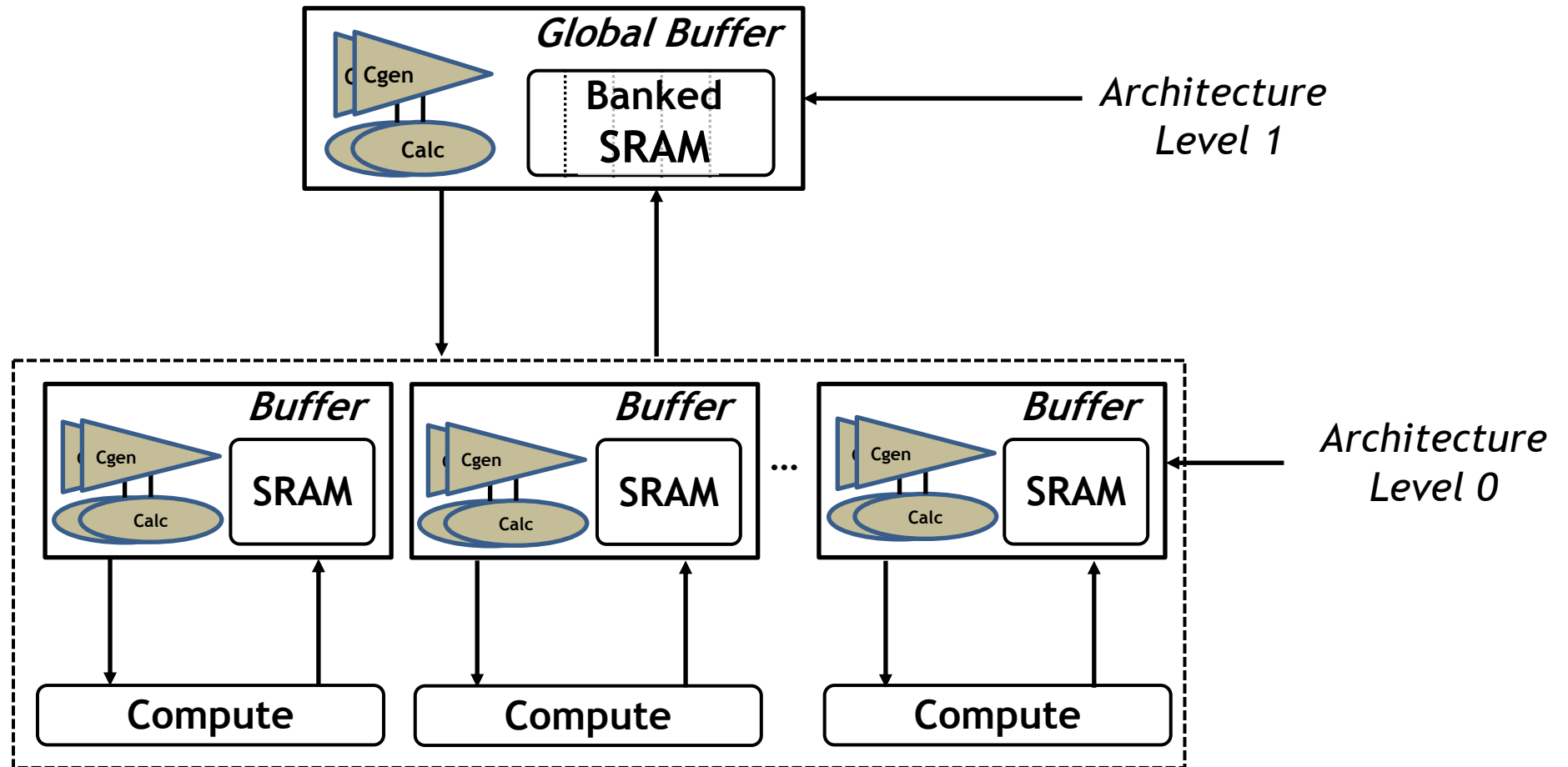


Fraction multiplication allows flexible representation of many sparsity degrees in a wide range

HSS Enables Modularized Acceleration

Modularity of HSS allows different architecture levels to accelerate for different HSS ranks

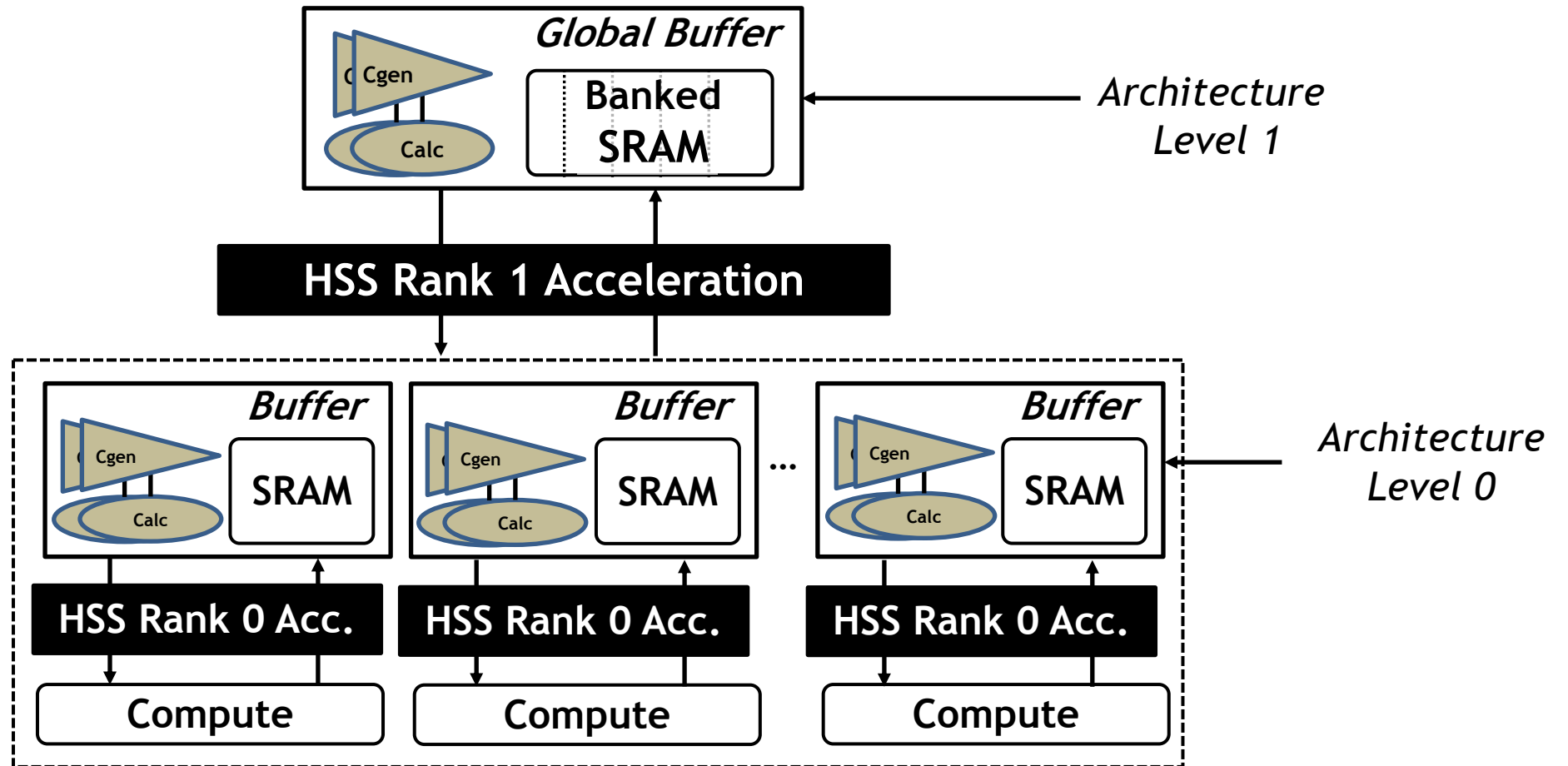
Example Accelerator Architecture Organization



HSS Enables Modularized Acceleration

Modularity of HSS allows different architecture levels to accelerate for different HSS ranks

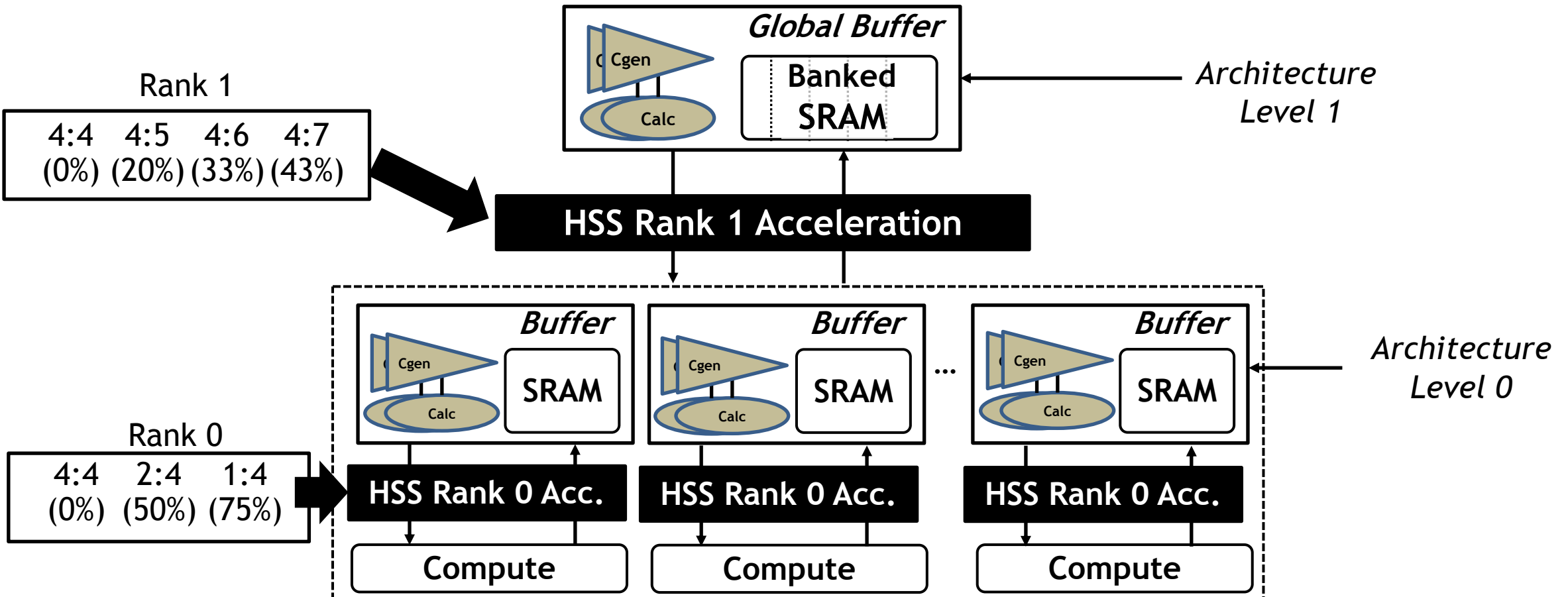
Example Accelerator Architecture Organization



HSS Enables Modularized Acceleration

Modularity of HSS allows different architecture levels to accelerate for different HSS ranks

Example Accelerator Architecture Organization

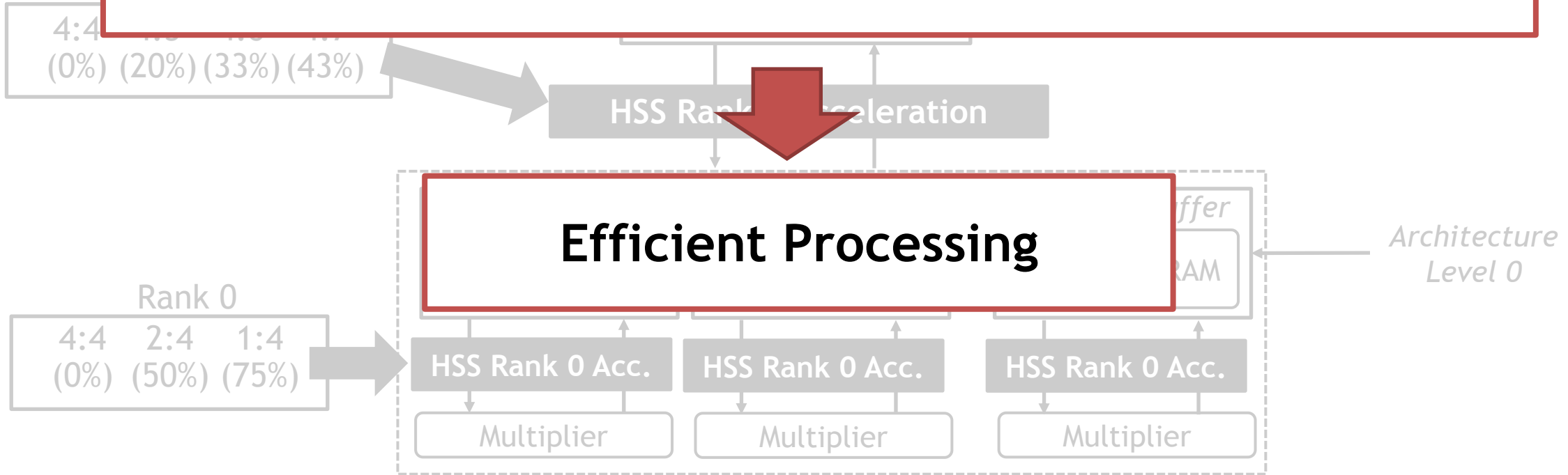


Each level only needs to accelerate for a few sparsity degrees

HSS Enables Modularized Acceleration

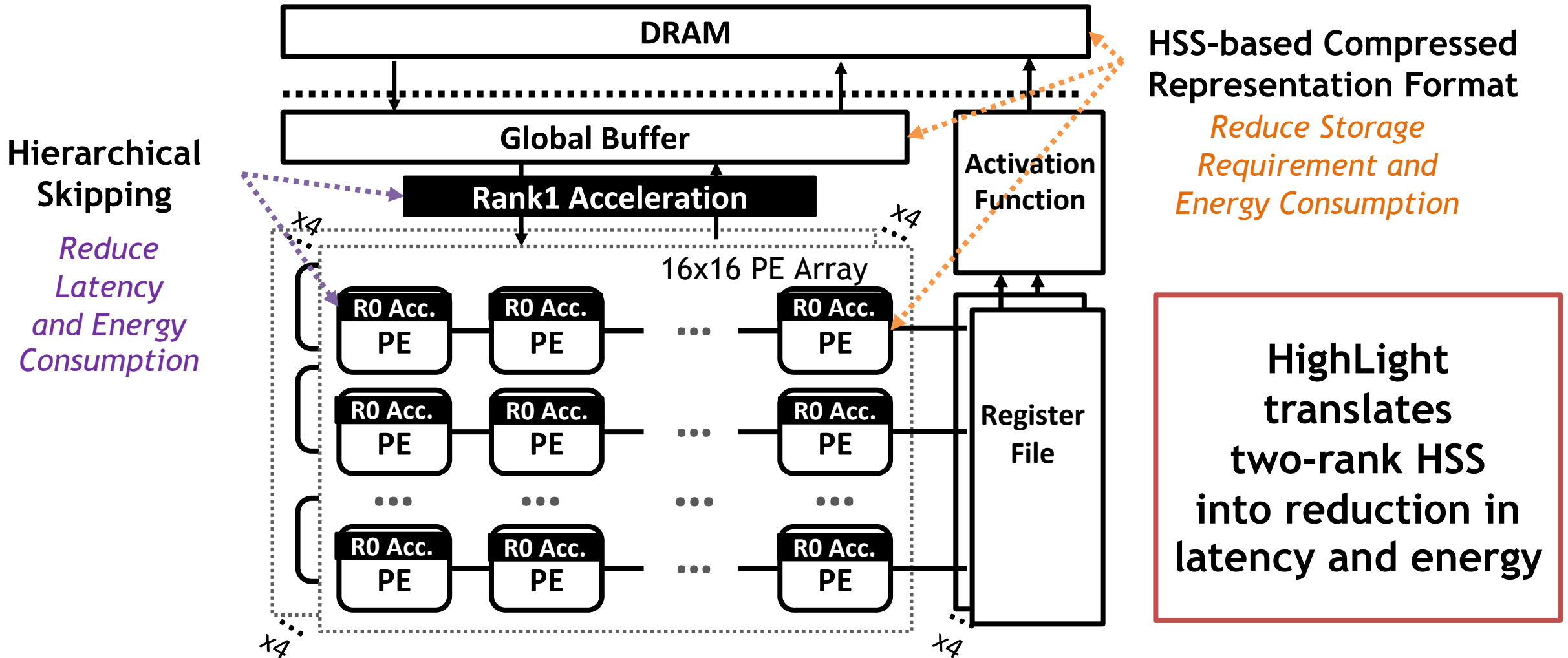
Modularity of HSS allows different architecture levels to accelerate for different HSS ranks

Simple Acceleration at Each Architecture Level Leads to Low Hardware Overhead



Each level only needs to accelerate for a few sparsity degrees

HighLight: Flexible and Efficient Sparse DNN Accelerator



Experimental Results

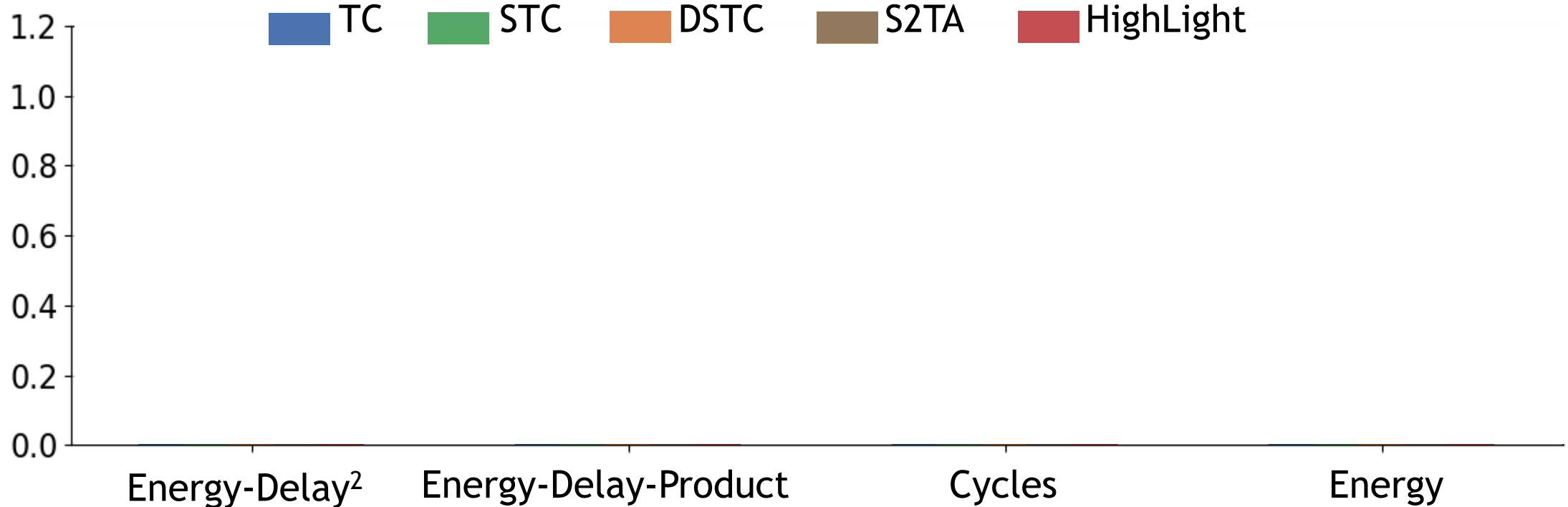
We Compare HighLight with Representative Designs



Geomean Across Various Hardware Performance Metrics

We evaluate the designs with synthetic workloads with different sparsity degrees ranging from 0%-75%

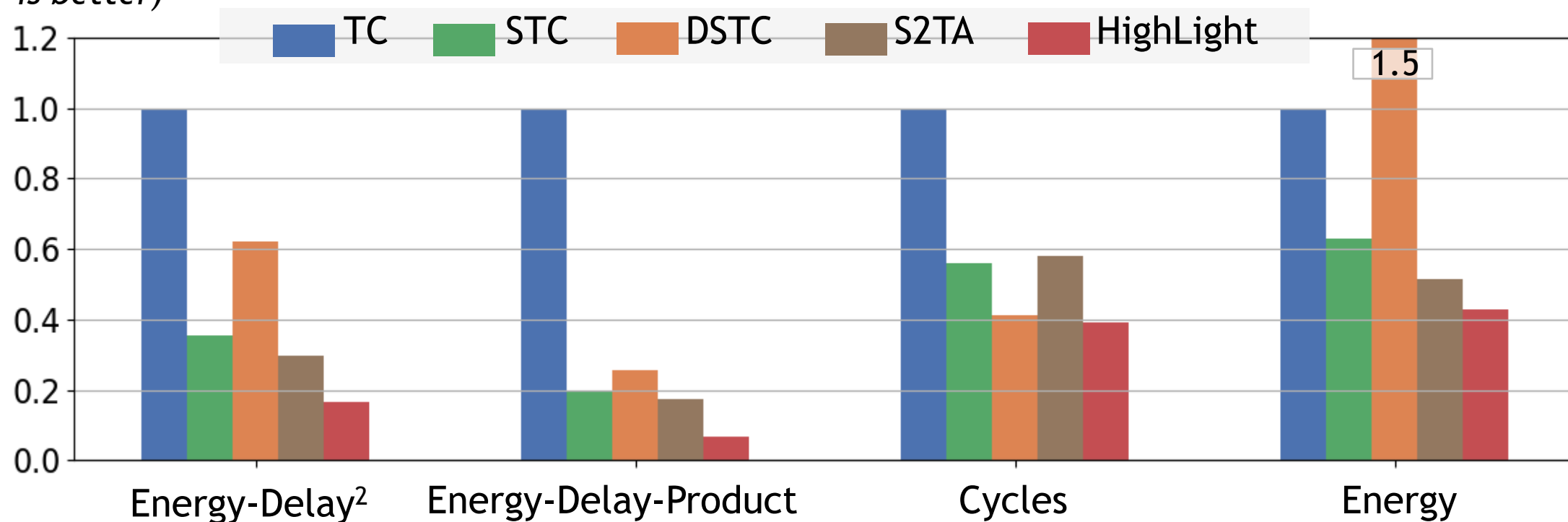
Geomean
(lower is better)



Geomean Across Various Hardware Performance Metrics

We evaluate the designs with synthetic workloads with different sparsity degrees ranging from 0%-75%

Geomean
(lower is better)



HighLight is efficient for all evaluated metrics

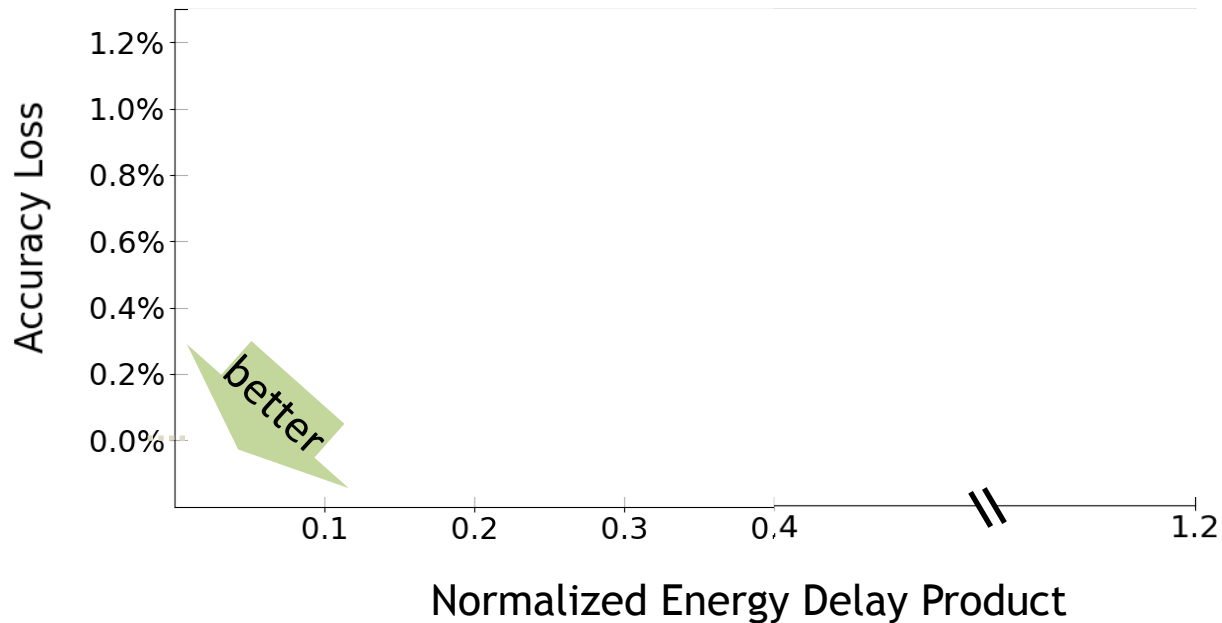
Accuracy-Energy Delay Product Pareto Frontier

We evaluate the designs with representative DNNs pruned to different sparsity degrees, each with its respective sparsity structure (if any)

Accuracy-Energy Delay Product Pareto Frontier

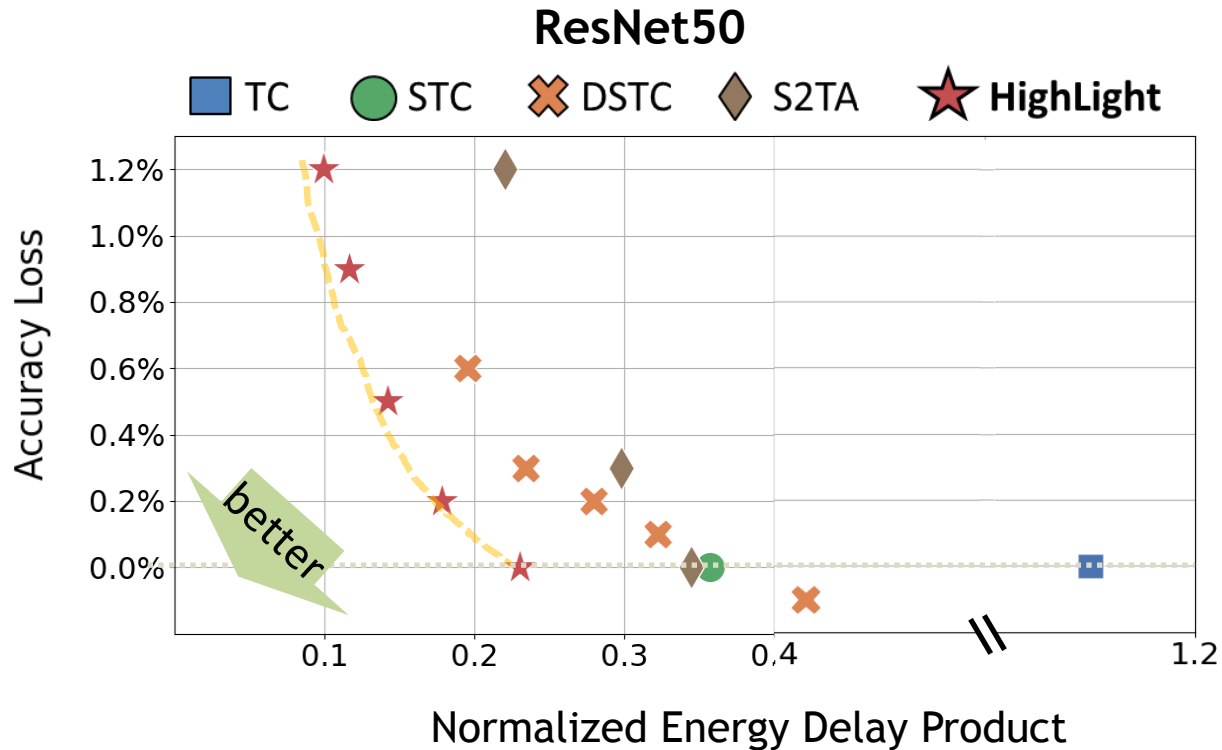
We evaluate the designs with representative DNNs pruned to different sparsity degrees, each with its respective sparsity structure (if any)

ResNet50



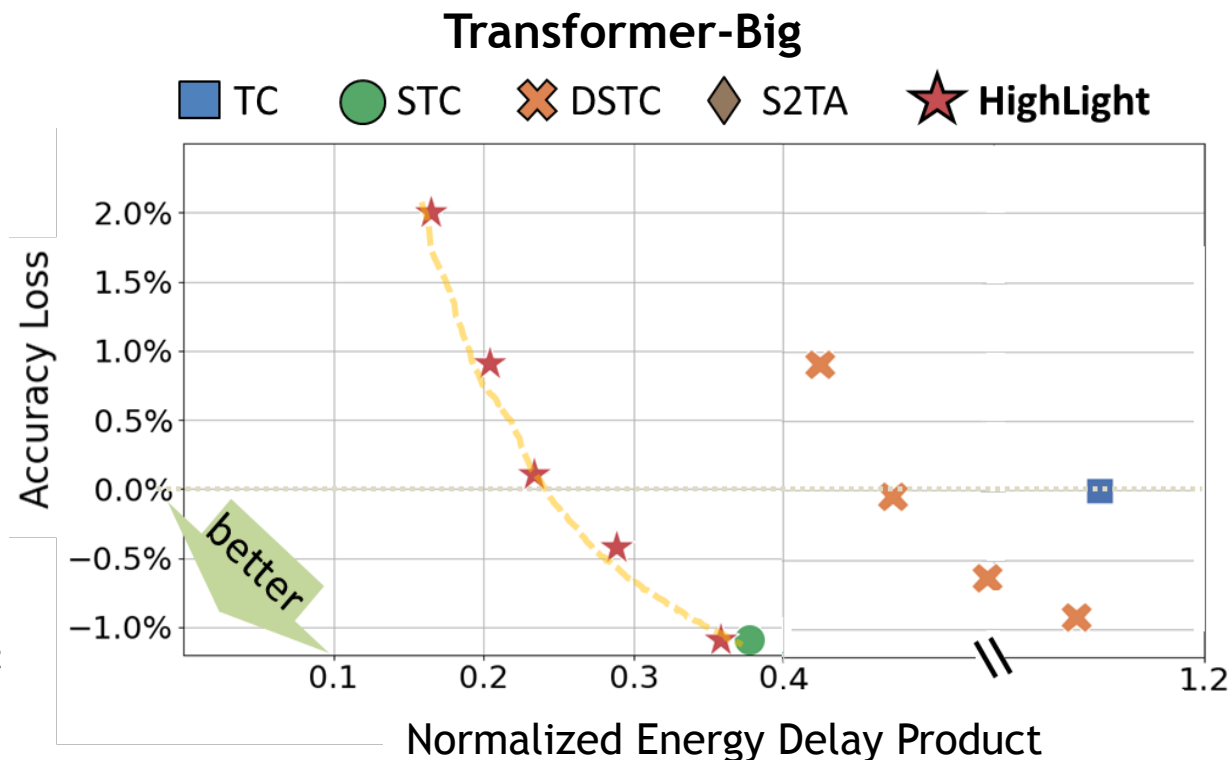
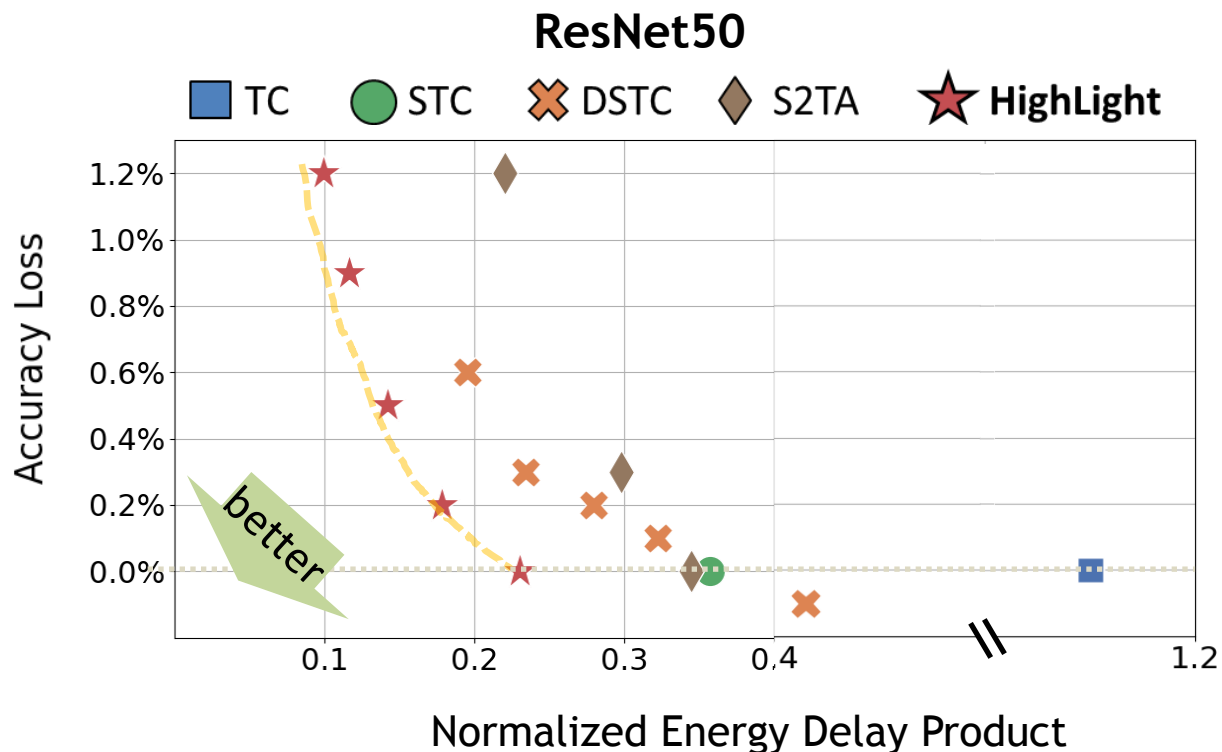
Accuracy-Energy Delay Product Pareto Frontier

We evaluate the designs with representative DNNs pruned to different sparsity degrees, each with its respective sparsity structure (if any)



Accuracy-Energy Delay Product Pareto Frontier

We evaluate the designs with representative DNNs pruned to different sparsity degrees, each with its respective sparsity structure (if any)



HighLight sits on the accuracy-energy delay product pareto frontier

HSS/HighLight Summary

Hierarchical Structured Sparsity (HSS)

- Composed of multiple levels of simple sparsity patterns
- Allows flexible expression of diverse sparsity degrees

HighLight Accelerator

- Supports two-rank HSS for a few degrees at each level
- Implements low-overhead support for each rank at different architecture levels
- Ensures both efficiency and flexibility

HSS/HighLight Summary

Hierarchical Structured Sparsity (HSS)

- Composed of multiple levels of simple sparsity patterns
- Allows flexible expression of diverse sparsity degrees

HighLight Accelerator

- Supports two-rank HSS for a few degrees at each level
- Implements low-overhead support for each rank at different architecture levels
- Ensures both efficiency and flexibility
- Outperforms existing works in terms of various hardware performance metrics
- Sits on the accuracy-performance pareto frontier for representative DNNs

Key Thesis Takeaway

- **Provided consistent design descriptions based on a systematic taxonomy, which facilitates communication.**
- **Demonstrated flexible, fast, and accurate modeling, which is key to early-stage design evaluation and exploration.**
- **Proposed DNN sparsity pattern and hardware co-design to ensure flexibility and efficiency, which is critical to next-generation accelerator designs.**