Sparse Tensor Accelerators: Abstraction and Modeling

Background Lecture Part 2

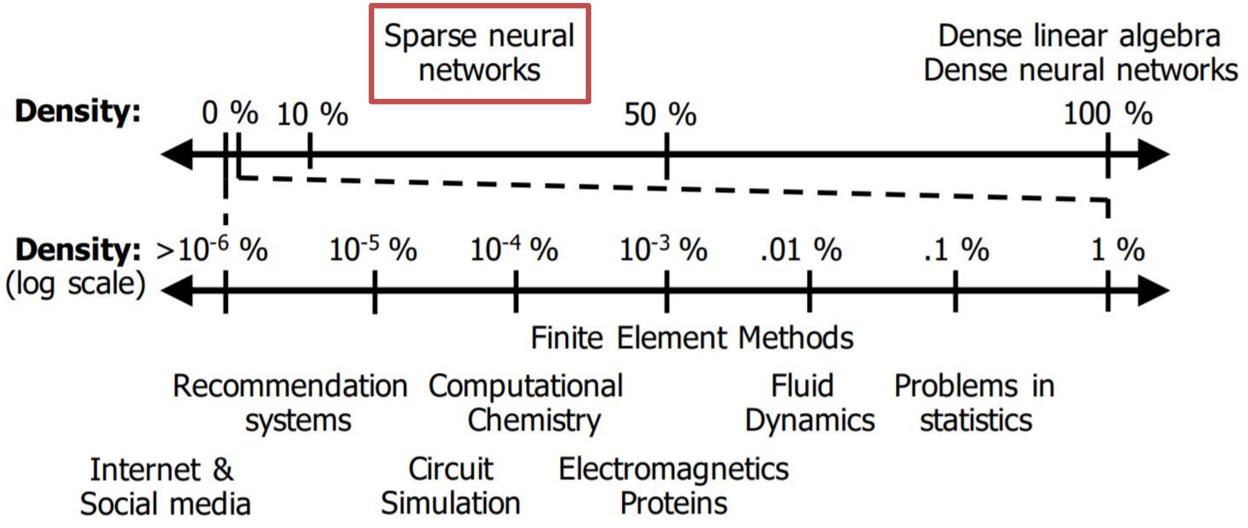
Joel Emer
Angshuman Parashar
Vivienne Sze
Po-An Tsai
Nellie Wu

ISCA Tutorial

June 2021

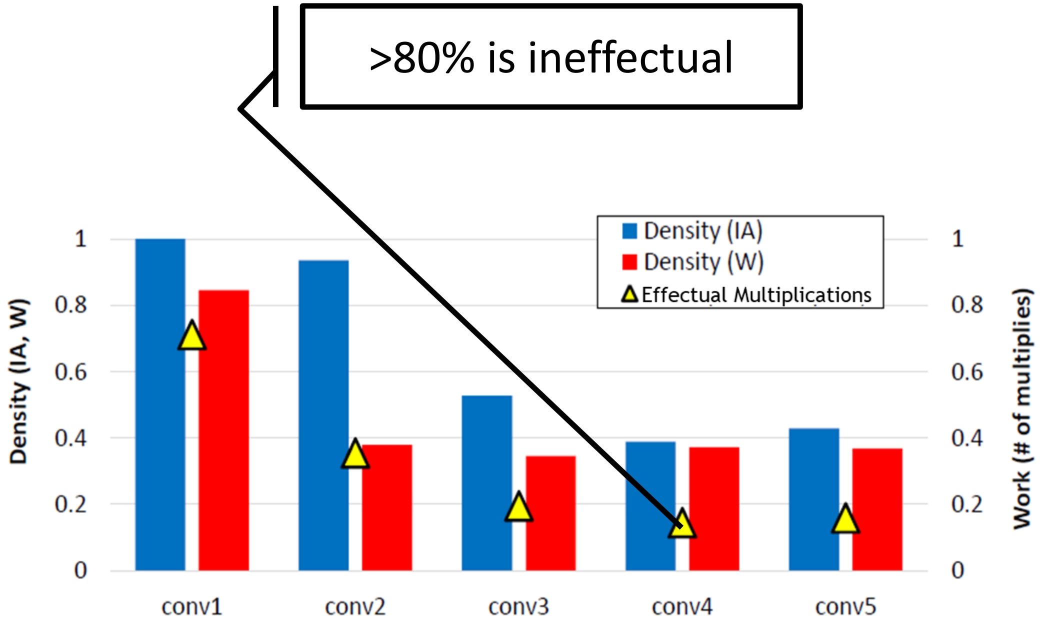
NVIDIA
Massachusetts Institute of Technology

Sparse Tensor Algebra in Popular Applications



Workload Sparsity by Workload Domain

[Hedge, MICRO19]



Pruned AlexNet Density

[Parashar, ISCA17]

0 x Anything = 0

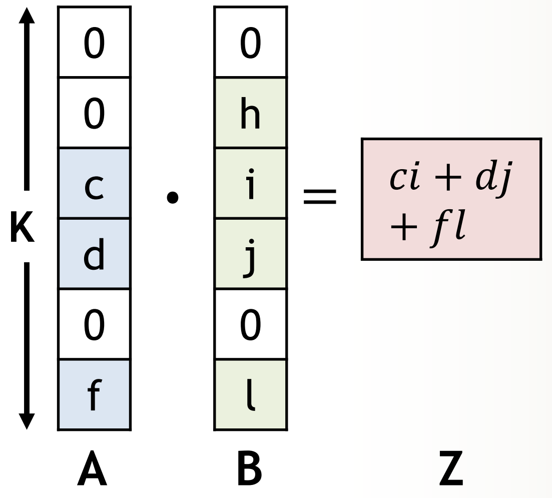
0 + Anything = Anything

Ineffectual Computations

Processing Uncompressed Sparse Tensor Workloads

Example Workload: Dot Product of Vectors

$$Z = \sum_{k=0}^{K} A[k] * B[k]$$

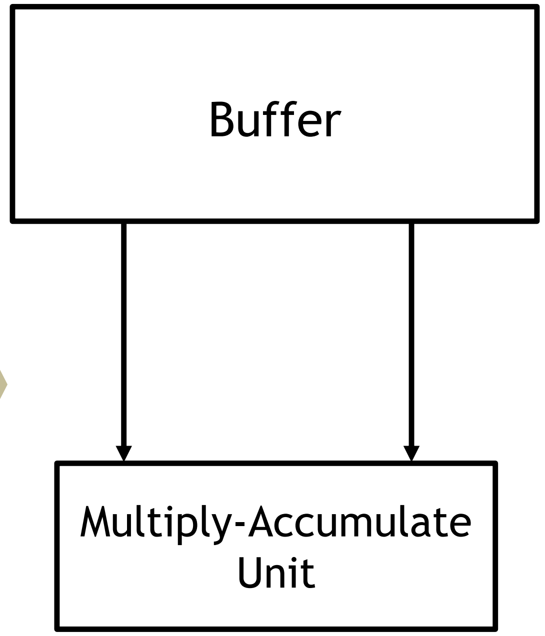


Mapping

Scheduling of data movement & compute in time & space

```
for k in [0:K)
  Z += A[k] * B[k]
```

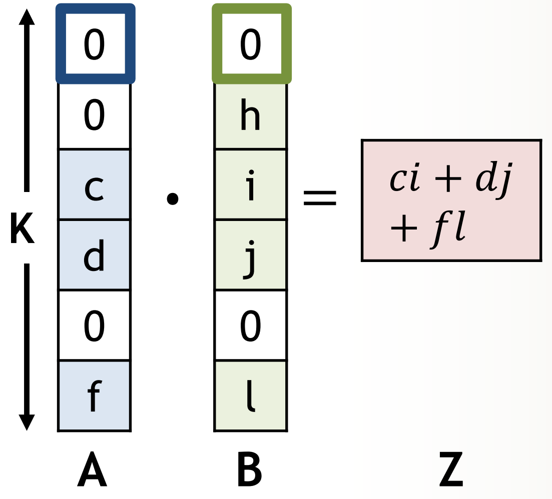
Accelerator Architecture



Processing Uncompressed Sparse Tensor Workloads

Example Workload: Dot Product of Vectors

$$Z = \sum_{k=0}^{K} A[k] * B[k]$$

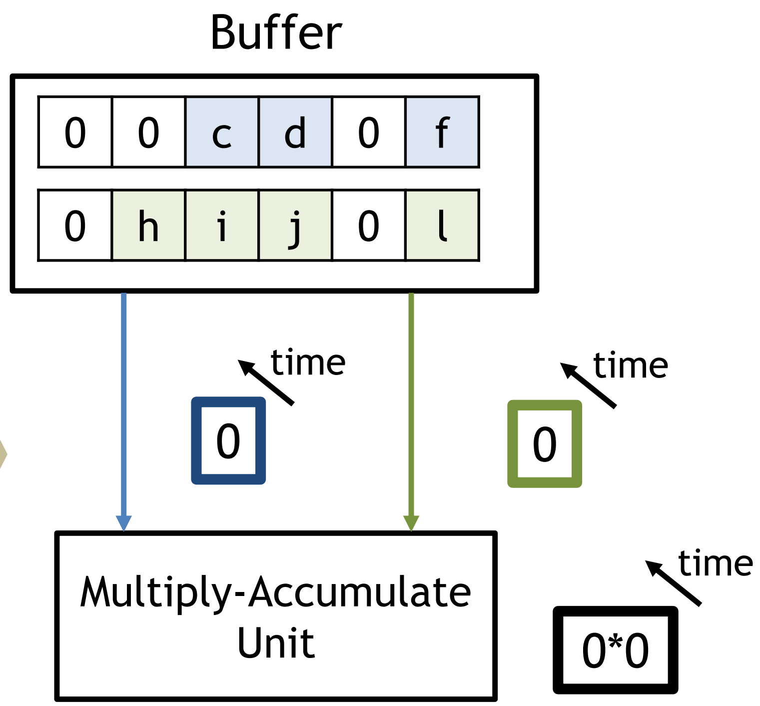


Mapping

Scheduling of data movement & compute in time & space

```
for k in [0:K)
  Z += A[k] * B[k]
```

Accelerator Architecture

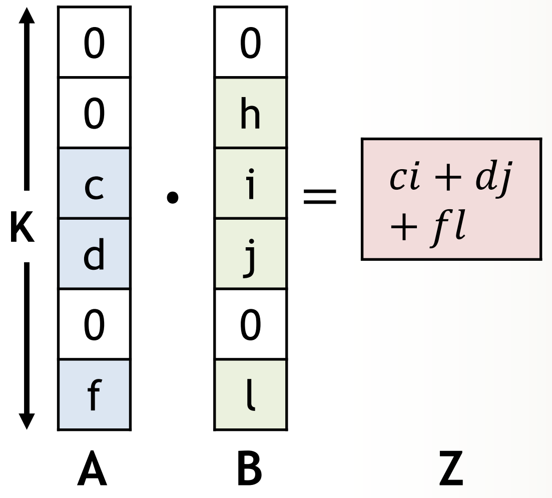


**Z data movements not shown*

Processing Uncompressed Sparse Tensor Workloads

Example Workload: Dot Product of Vectors

$$Z = \sum_{k=0}^{K} A[k] * B[k]$$

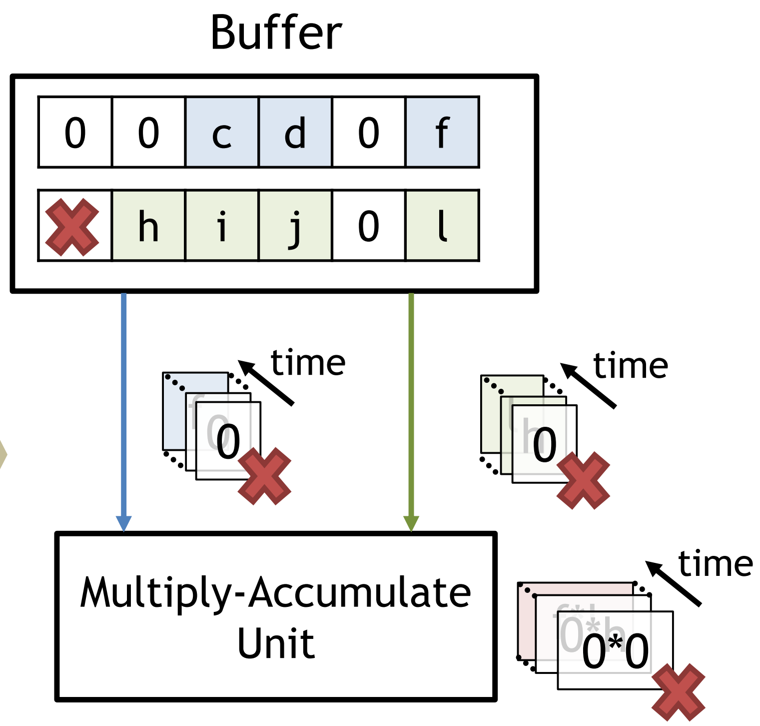


Mapping

Scheduling of data movement & compute in time & space

```
for k in [0:K)
  Z += A[k] * B[k]
```

Accelerator Architecture



**Z data movements not shown*

Ineffectual computations introduce opportunities to exploit zero-based savings in hardware

Hardware Sparse Optimization Features

Format:
Choose tensor representations to save necessary storage spaces and energy associated zero accesses

Gating:
Explicitly eliminate ineffectual storage accesses and computes by letting the hardware unit staying idle for the cycle to save energy

Skipping:
Explicitly eliminate ineffectual storage accesses and computes by skipping the cycle to save energy and time

Various Implementations Lead to Different Performance

Format:
Choose tensor representations to save necessary storage spaces and energy associated zero accesses

Gating:
Explicitly eliminate ineffectual storage accesses and computes by letting the hardware unit staying idle for the cycle to save energy

Skipping:
Explicitly eliminate ineffectual storage accesses and computes by skipping the cycle to save energy and time

What is the chosen format?

Do all tensors share the same format?

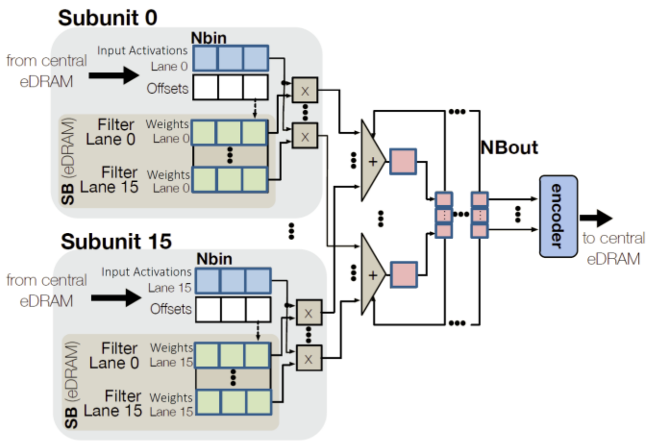
When is a storage access gated?

How much is the compute able to skip ahead?

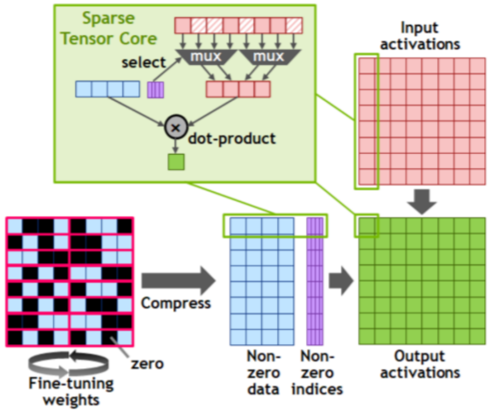
At which storage level is the skipping performed?

What is the criteria for skipping?

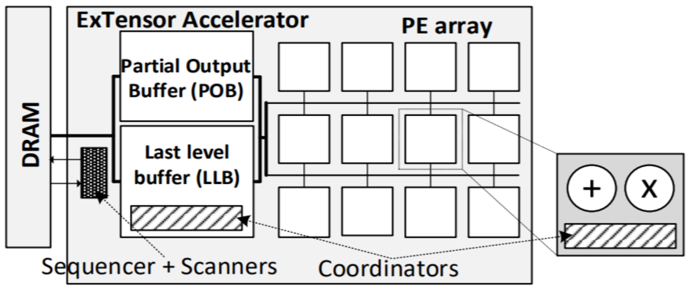
Diverse Sparse Tensor Accelerator Designs



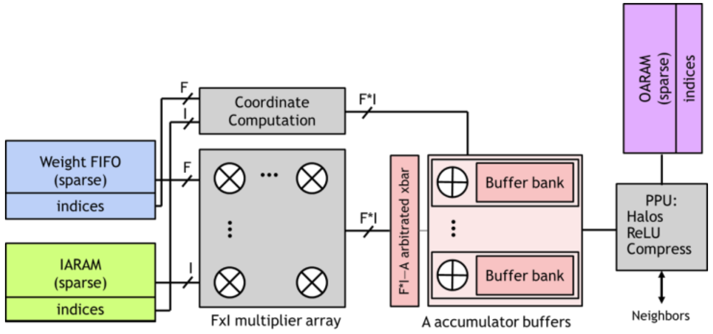
Cnvlutin [ISCA2016]



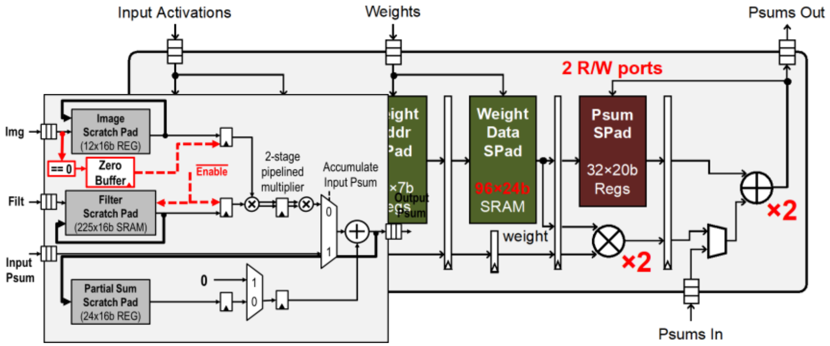
Tensor Core V3 [NVIDIA2020]



ExTensor [MICRO2019]



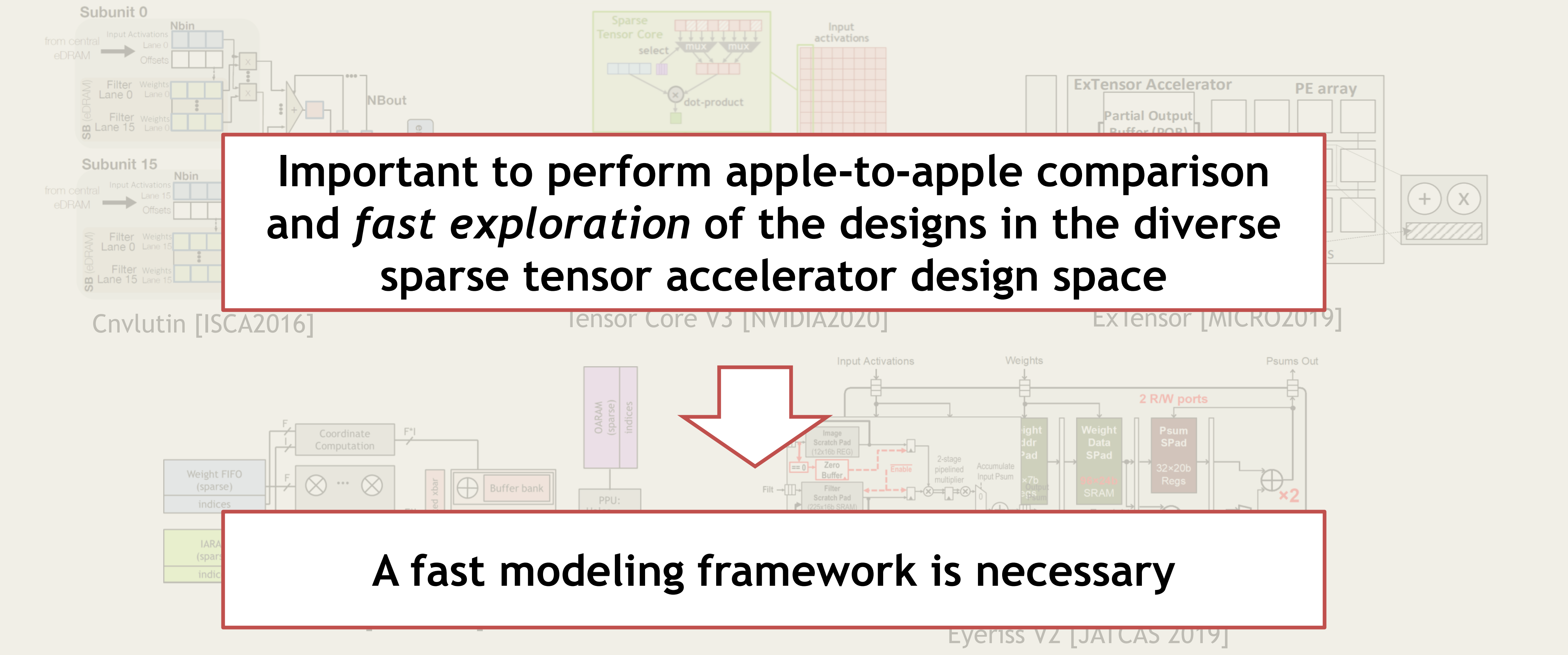
SCNN [ISCA2017]



Eyeriss V1 [JSSC 2017]
Eyeriss V2 [JATCAS 2019]

Each accelerator design carefully combines sparse optimization features that work the best with its architecture topology to improve energy efficiency and processing time

Diverse Sparse Tensor Accelerator Designs

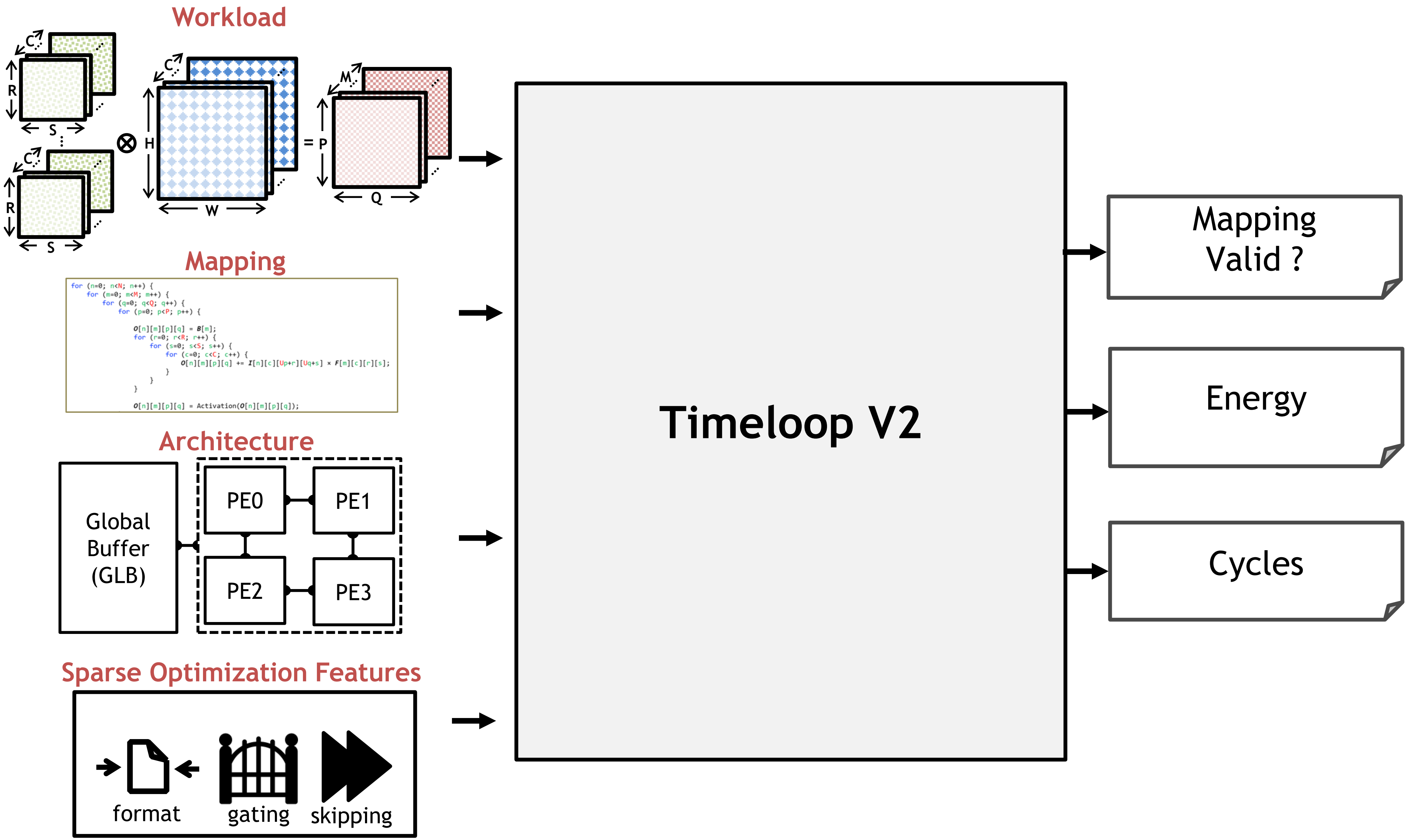


Important to perform apple-to-apple comparison and *fast exploration* of the designs in the diverse sparse tensor accelerator design space

A fast modeling framework is necessary

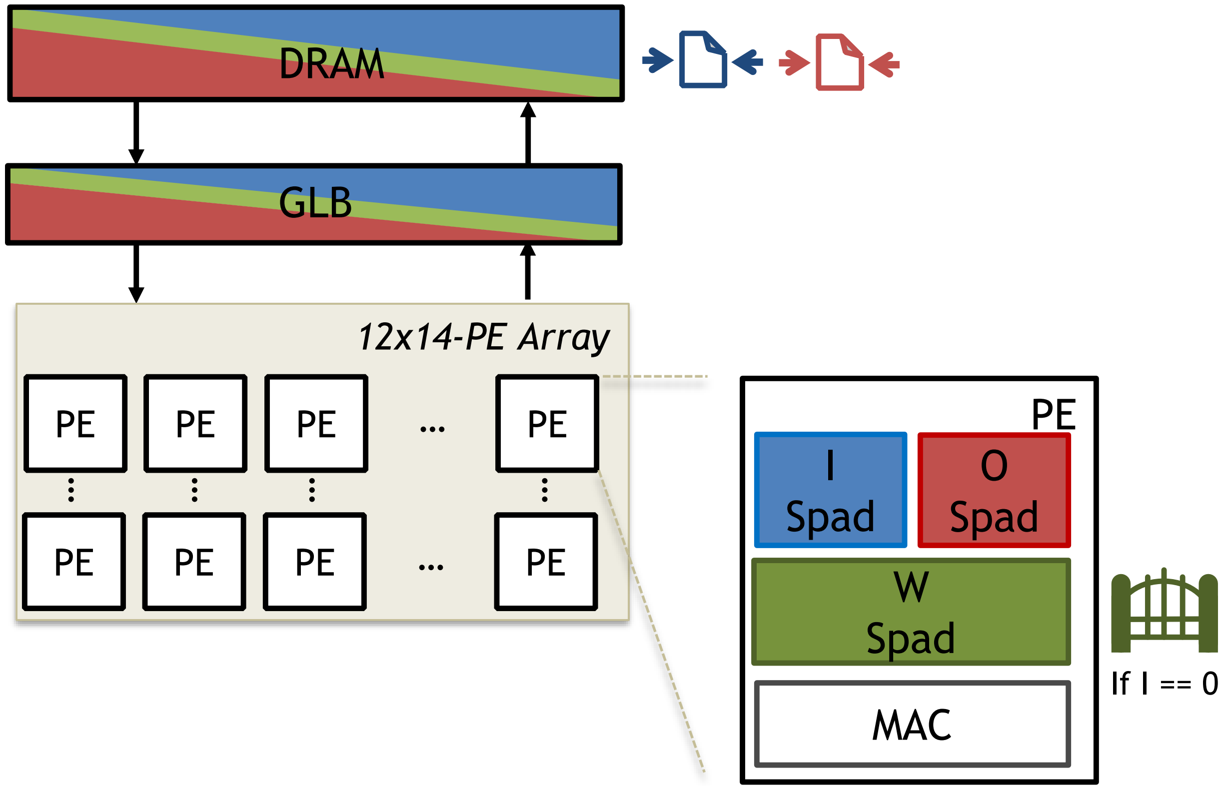
Each accelerator design carefully combines sparse optimization features that work the best with its architecture topology to improve energy efficiency and processing time

Analytical Sparse Tensor Accelerator Modeling



Validation on Eyeriss V1 [ISSCC 2016]

High-Level Architecture



Example Mapping (AlexNet Layer3)

Row Stationary Dataflow

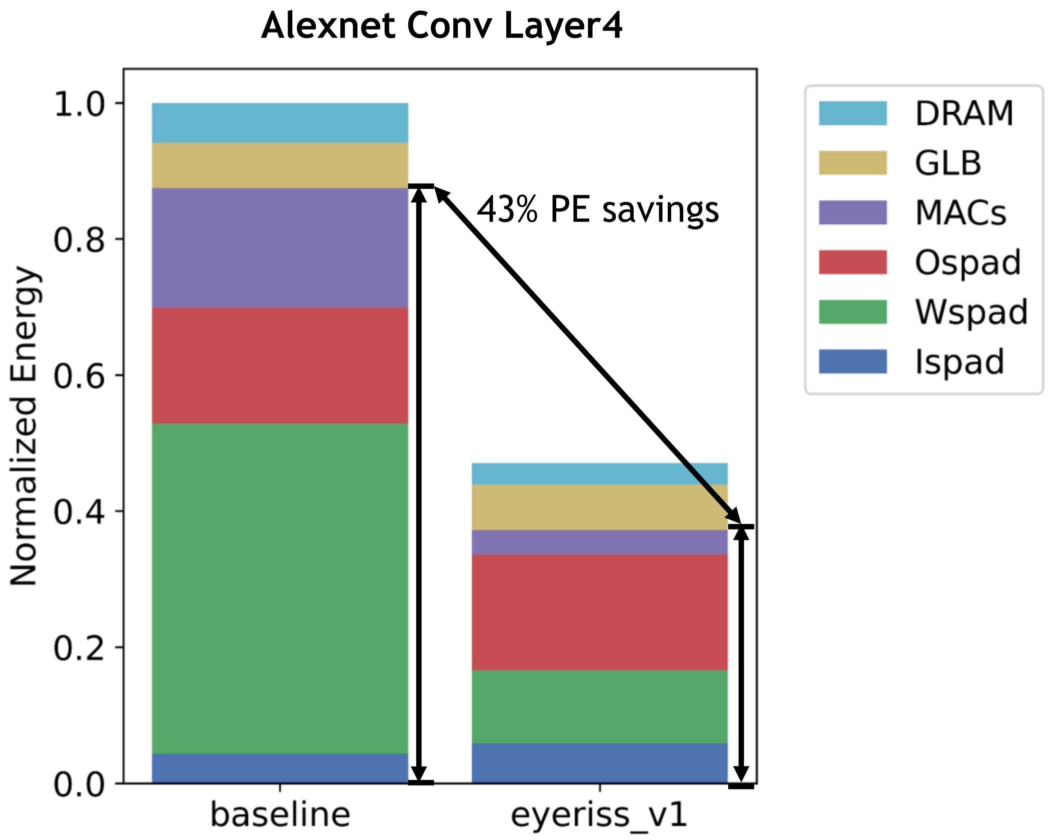
```
DRAM [ Weights:884736 (884736) Inputs:230400 (63361) Outputs:259584 (78654) ]
-----------------------------------------------------------------------------
| for M in [0:6)
|   for C in [0:64)

GLB [ Inputs:3600 (3600) Outputs:43264 (43264) ]
-------------------------------------------------------
|     for N in [0:4)
|       for P in [0:13)
|         for Q in [0:1)
|           for Q in [0:13) (Spatial-X)
|             for M in [0:4) (Spatial-Y)
|               for S in [0:3) (Spatial-Y)

ISpad[ Inputs:12 (12) ]
----------------------------
|                 for Q in [0:1)

WSpad [ Weights:192 (192) ]
---------------------------------
|                   for R in [0:3)
|                     for C in [0:4)

OSpad [ Outputs:16 (16) ]
----------------------------
|                       for M in [0:16)
```

Validation on Eyeriss V1 [ISSCC 2016]

- DRAM compression ratio

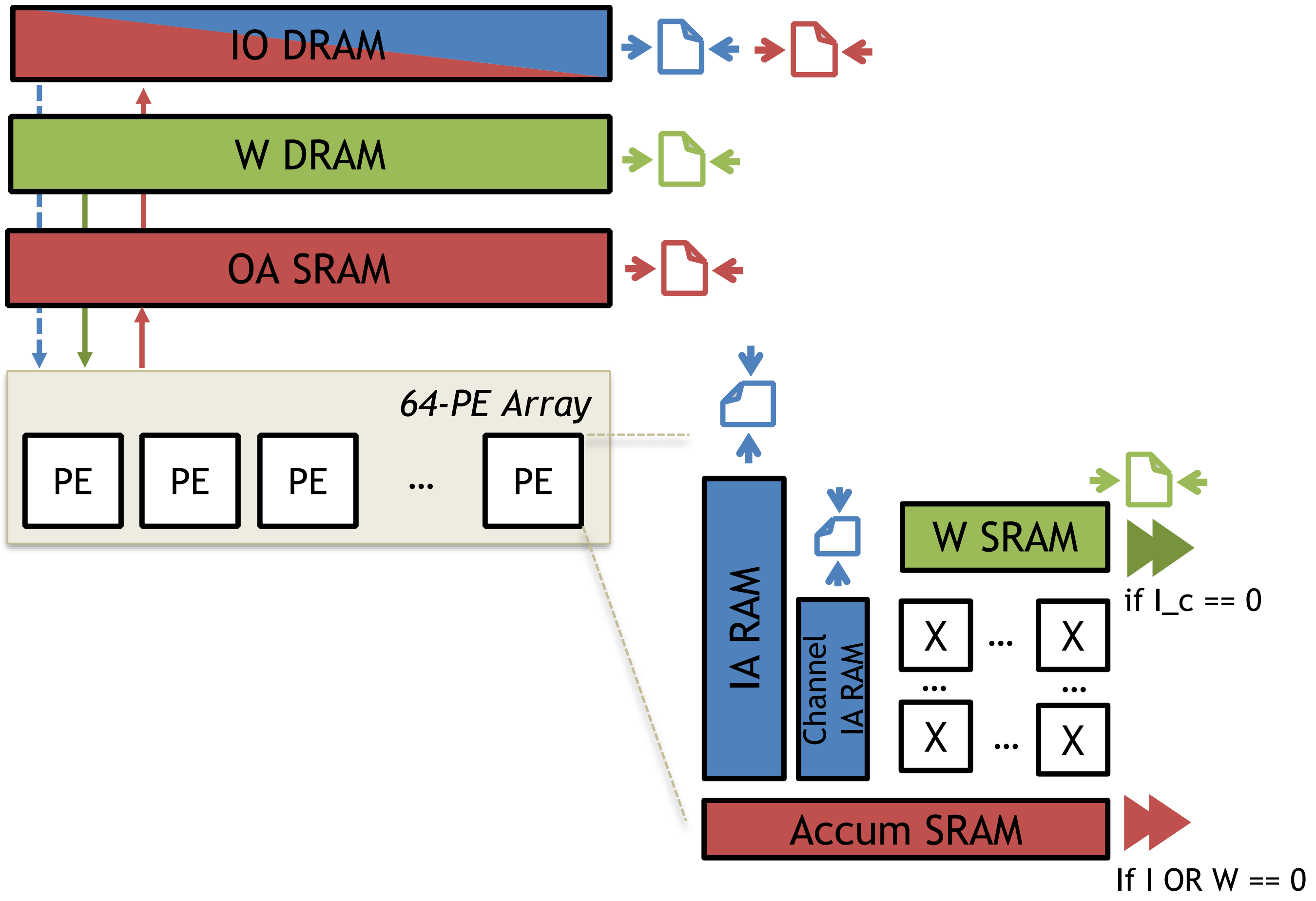
layer	Eyeriss	our work
1	1.2	1.24
2	1.4	1.37
3	1.7	1.68
4	1.8	1.86
5	1.9	1.93

- Normalized energy consumption with sparse optimization applied
  - 45% vs. 43% in our estimation, 96% accurate



Validation on SCNN Architecture [ISCA2017]

High-Level Architecture



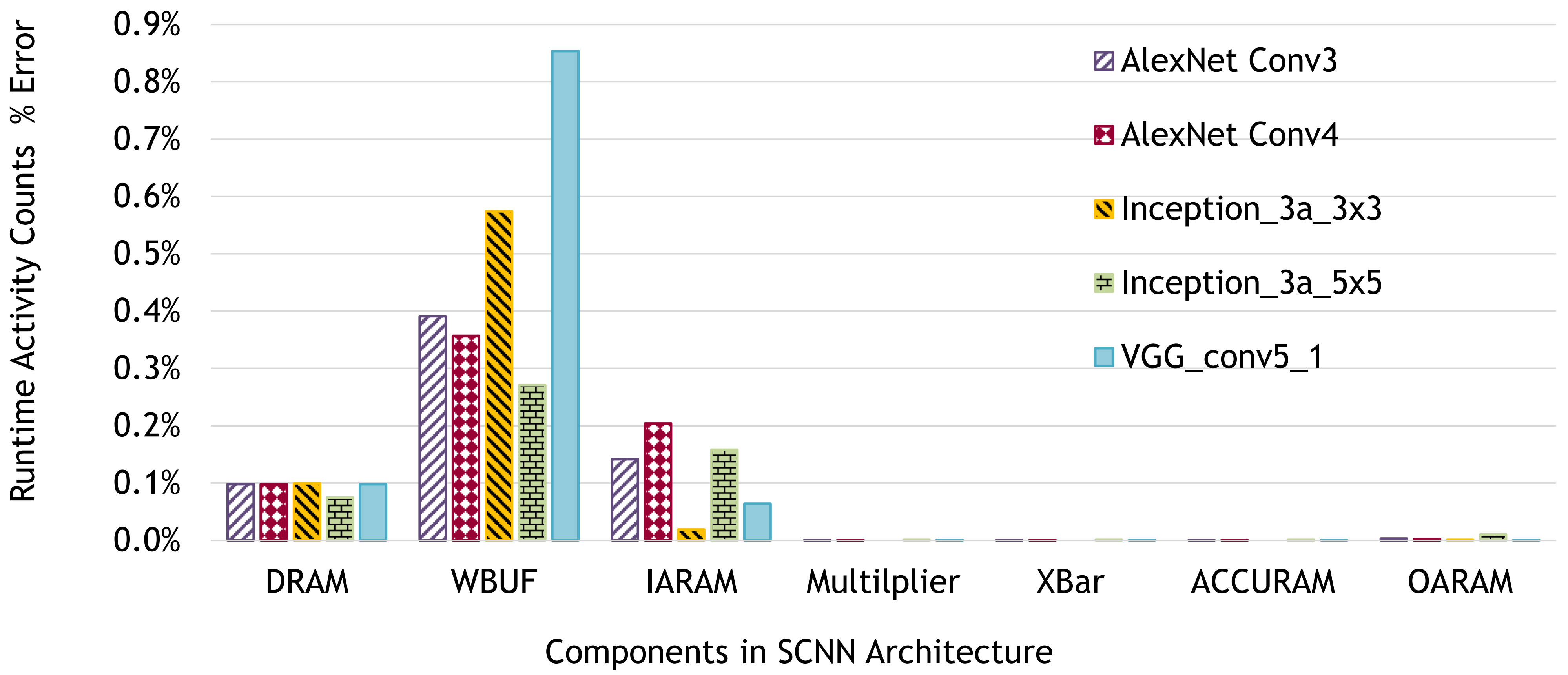
Example Mapping (AlexNet Layer3)

Input Stationary Cartisian Product Dataflow

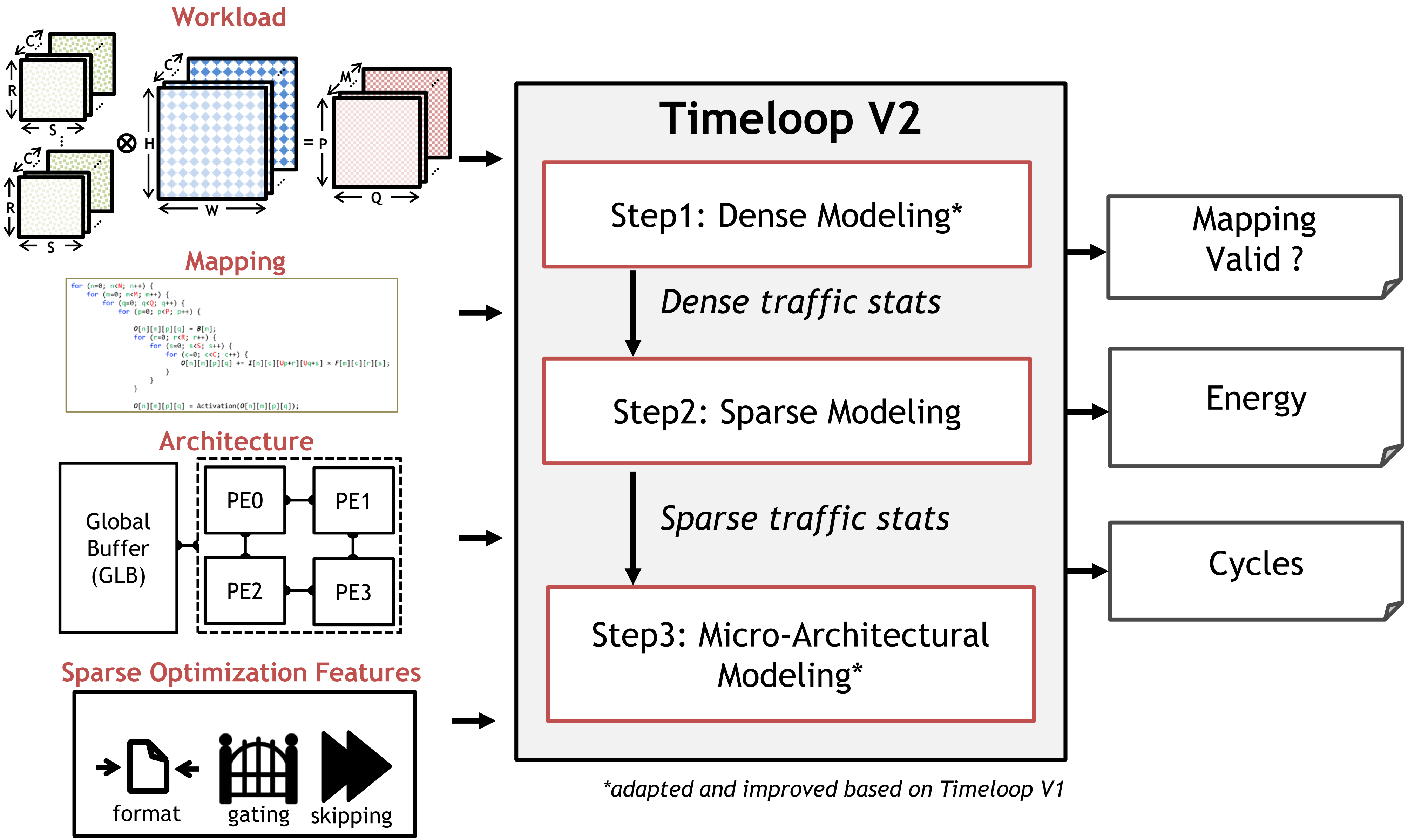
```
IO DRAM [ ]
----------
| for W in [0:1)

O ARAM [ Outputs:75264 (34742) ]
-------------------------------
|   for W in [0:1)

W DRAM [ Weights:884736 (325761) ]
---------------------------------
|     for M in [0:6)
|       for W in [0:6) (Spatial-X)
|         for H in [0:6) (Spatial-X)

IA RAM [ Inputs:1024 (639) ]
---------------------------
|           for W in [0:1)

Accumu SRAM [ Outputs:1024 (1024) ]
-----------------------------------------
|             for C in [0:256)

Channel IARAM [ Inputs:4 (4) ]
------------------------------
|               for W in [0:1)

W SRAM [ Weights:576 (213) ]
-----------------------------
|                 for M in [0:16)
|                   for S in [0:3)
|                     for R in [0:3)
|                       for M in [0:4) (Spatial-Y)
|                         for W in [0:2) (Spatial-X)
|                           for H in [0:2) (Spatial-X)
```

Validation on SCNN Architecture [ISCA2017]

Less than 1% error comparing to results generated by a custom SCNN simulator

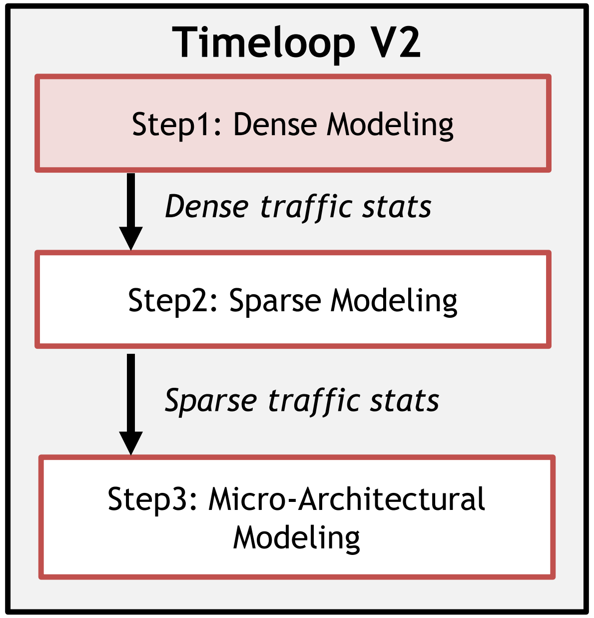


Proposed Analytical Sparse Tensor Accelerator Modeling



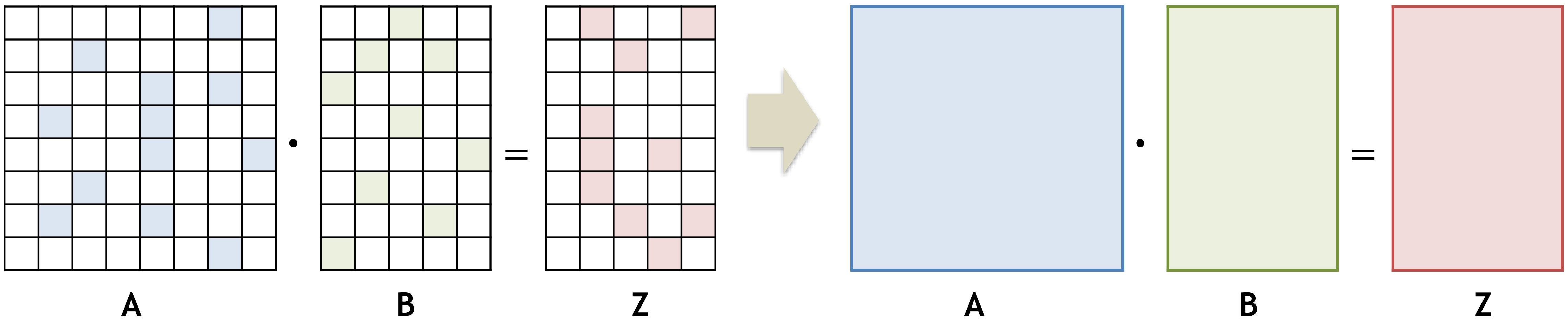
*adapted and improved based on Timeloop V1

Analytical Modeling for Dense Accelerators



Abstracts Problem Instance Details Away

Fast analytical modeling does not examine the exact data in workloads

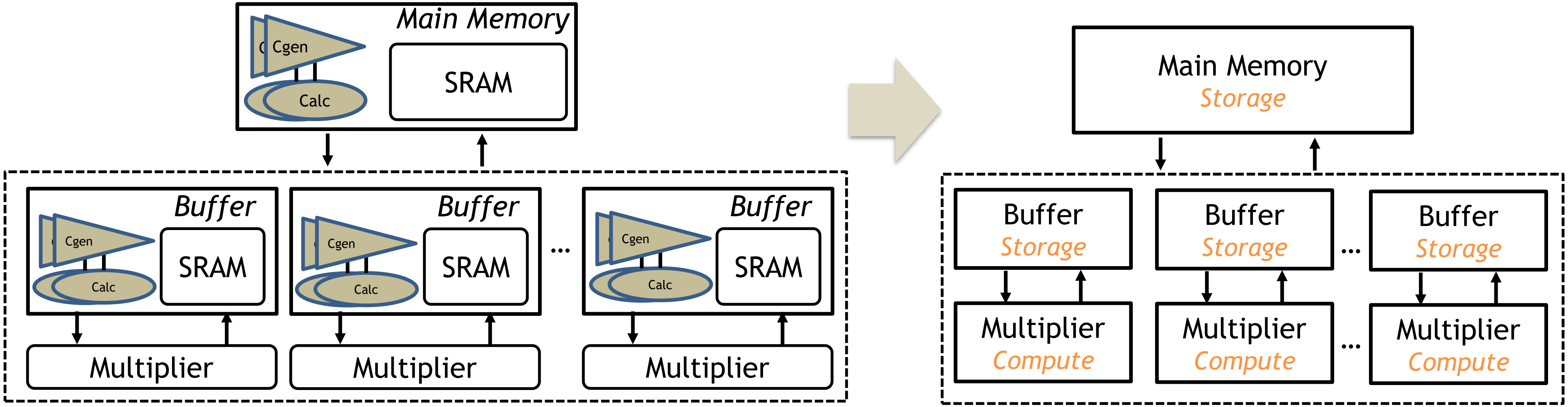


Exact Problem Instance

Problem Instance Shapes

Abstracts Architecture Details Away

Fast analytical modeling does not examine detailed architecture implementation



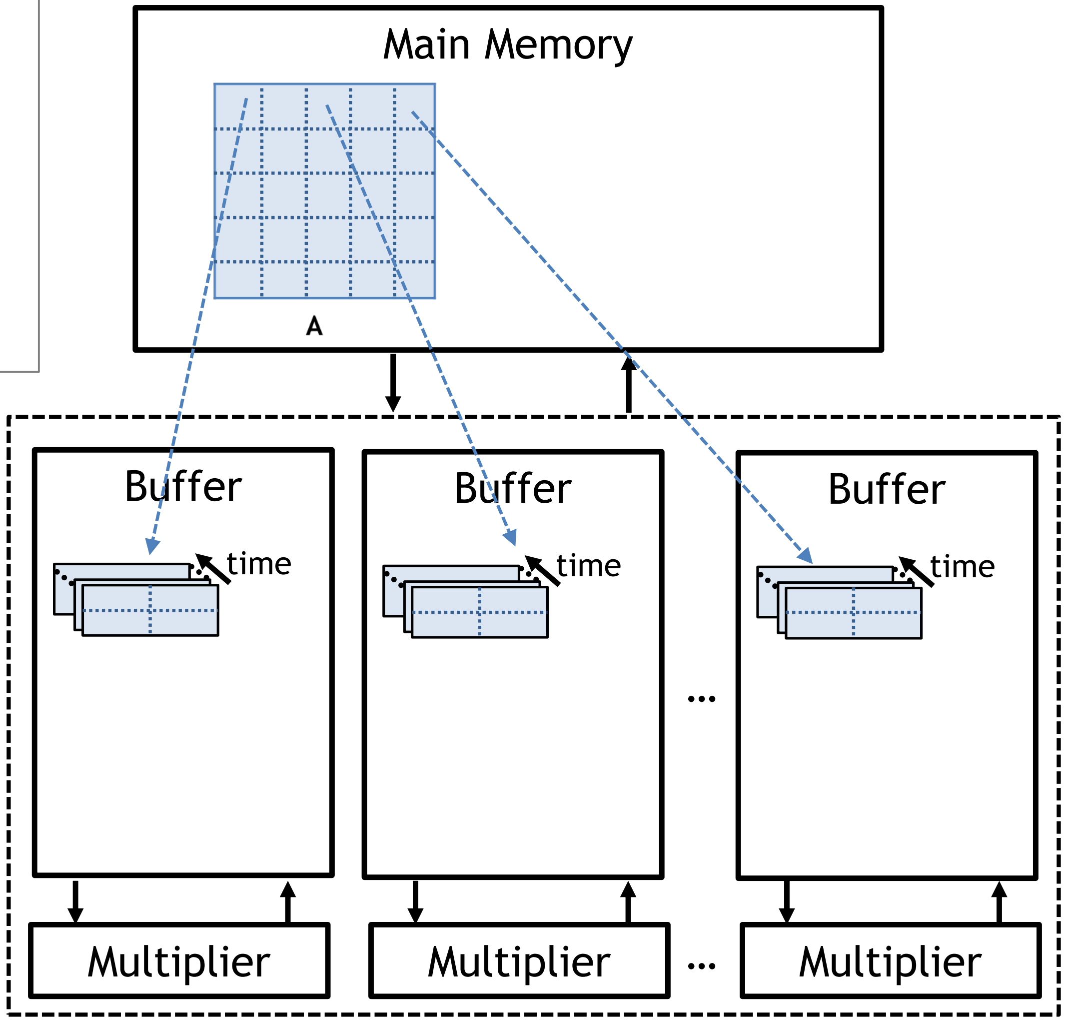
Detailed Architecture

Abstract Architecture Topology

Dense Data Movement and Compute Analysis

Example Mapping

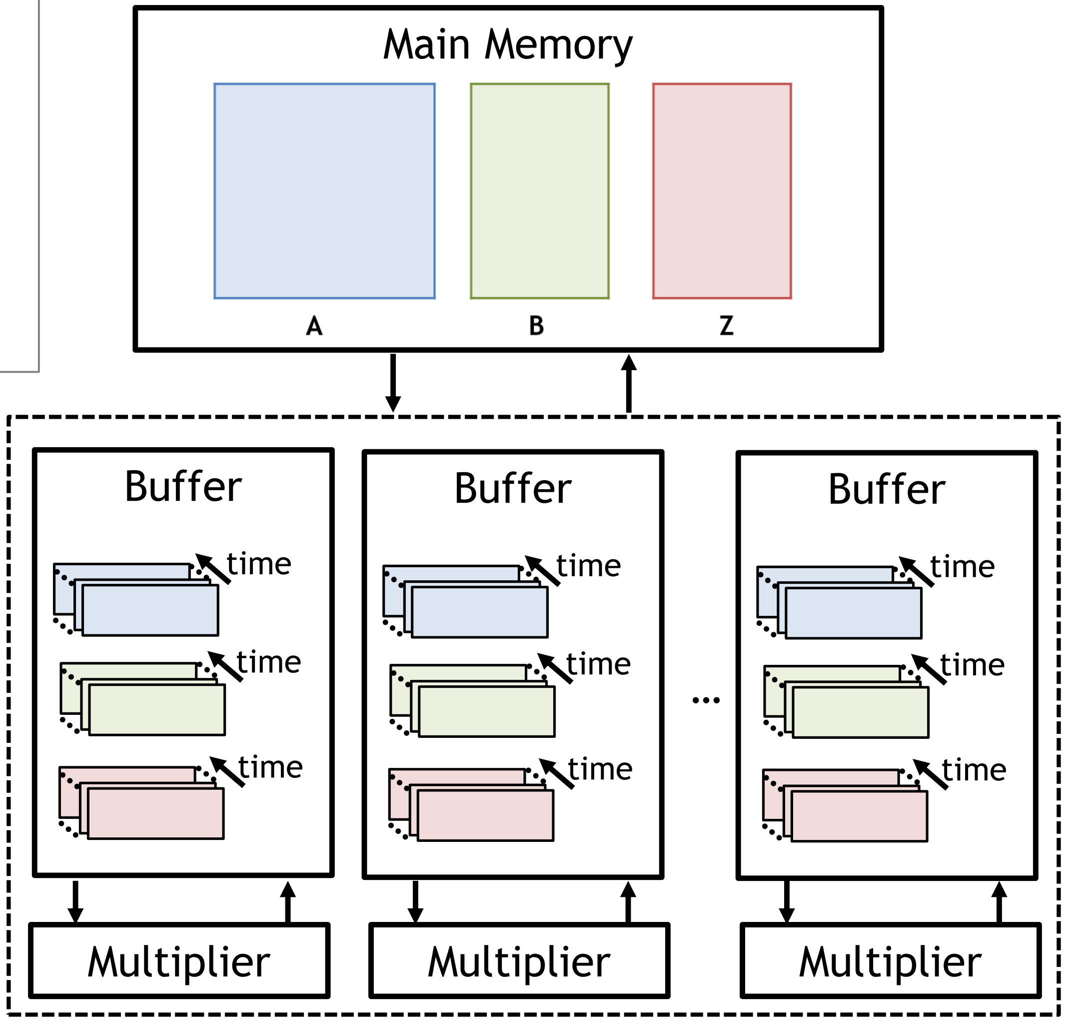
```
------ Main Memory -------
for m in [0:M2)
 for n in [0:N2)
  for k in [0:K2)
   par-for m in [0:M1)
   par-for n in [0:N1)
   par-for k in [0:K1)
------ Buffer-------
    for m in [0:M0)
     for n in [0:N0)
      for k in [0:K0)
       Z[m,n] += A[m,k]*B[k,n]
```



Dense Data Movement and Compute Analysis

Example Mapping

```
------ Main Memory -------
for m in [0:M2)
 for n in [0:N2)
  for k in [0:K2)
   par-for m in [0:M1)
   par-for n in [0:N1)
   par-for k in [0:K1)
------ Buffer-------
    for m in [0:M0)
     for n in [0:N0)
      for k in [0:K0)
       Z[m,n] += A[m,k]*B[k,n]
```



Answer dataflow related questions

- Which tensor is temporally reused at each storage level?
- How much data is transferred between storages?
- How many compute happened?
- ...

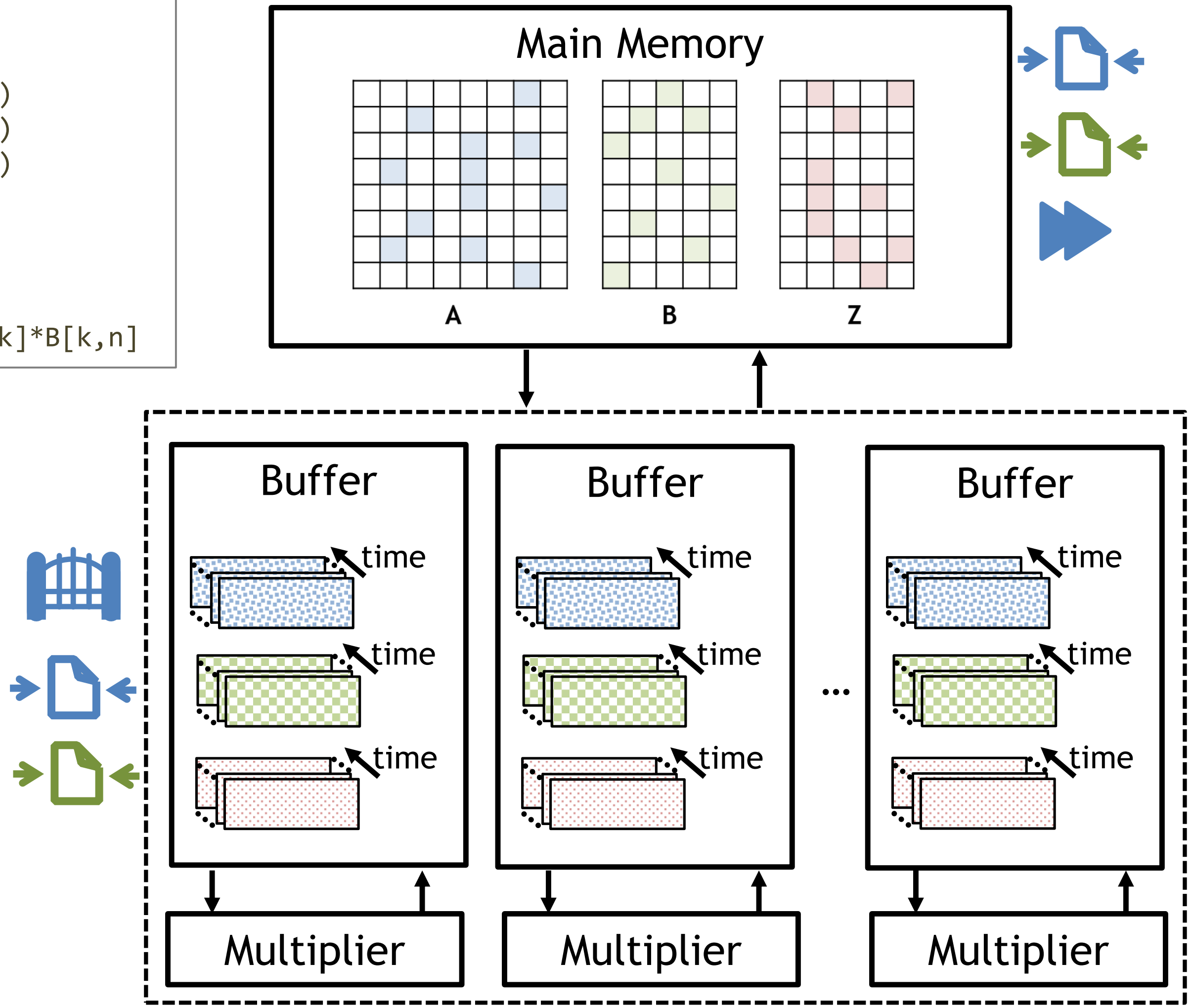
Mapping Valid?
Energy Efficiency
Cycles

* *More detailed explanation of the dense analysis can be found in Timeloop [Parashar, ISPASS 2019]*

Sparse Accelerator Modeling is Data Dependent

Example Mapping

```
------ Main Memory -------
for m in [0:M2)
 for n in [0:N2)
  for k in [0:K2)
   par-for m in [0:M1)
   par-for n in [0:N1)
   par-for k in [0:K1)
------ Buffer-------
    for m in [0:M0)
     for n in [0:N0)
      for k in [0:K0)
       Z[m,n] += A[m,k]*B[k,n]
```



What is impact of sparse optimization features?

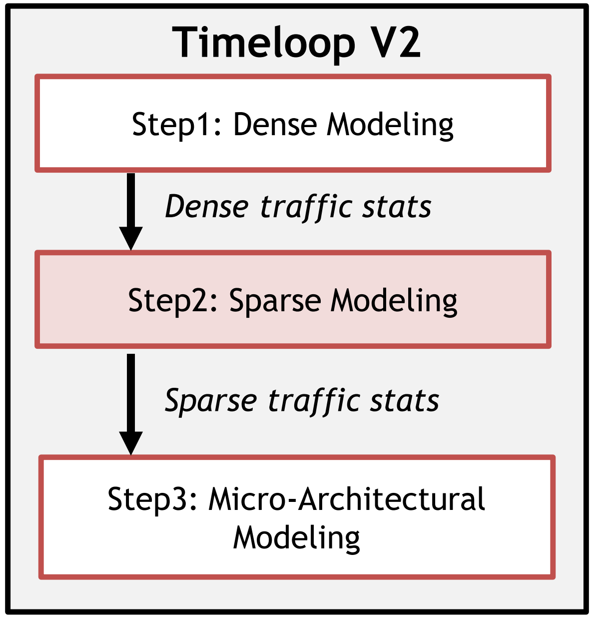
Answer dataflow related questions

- Which tensor is temporally reused at each storage level?
- How much data is transferred between storages?
- How many compute happened?
- …

Mapping Valid?
Energy Efficiency
Cycles

* *More detailed explanation of the dense analysis can be found in Timeloop [Parashar, ISPASS 2019]*

Proposed Sparse Tensor Accelerator Modeling Methodology



Specifications and Their Interactions

Mapping

Statistical Workload Density Models

Sparse Optimization Features

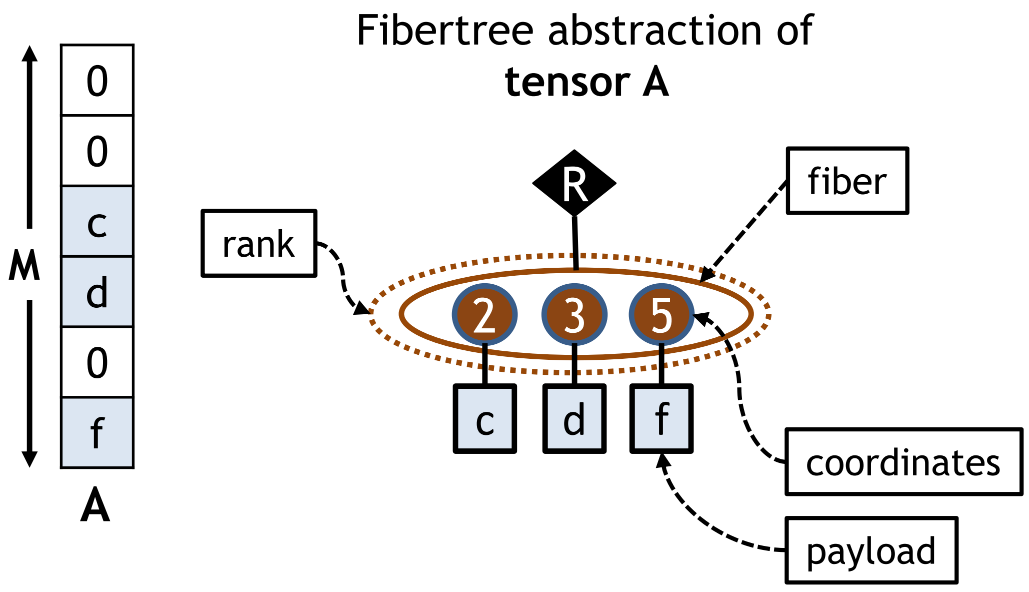
Interactions

- Required hardware capacity at each level
- Resulting sparse traffic of the storage and compute units

Proposed Sparse Tensor Accelerator Modeling Methodology

Interactions Between Mapping and Workload Density Models

Analysis Based on Fibertree-based Tensor Abstraction

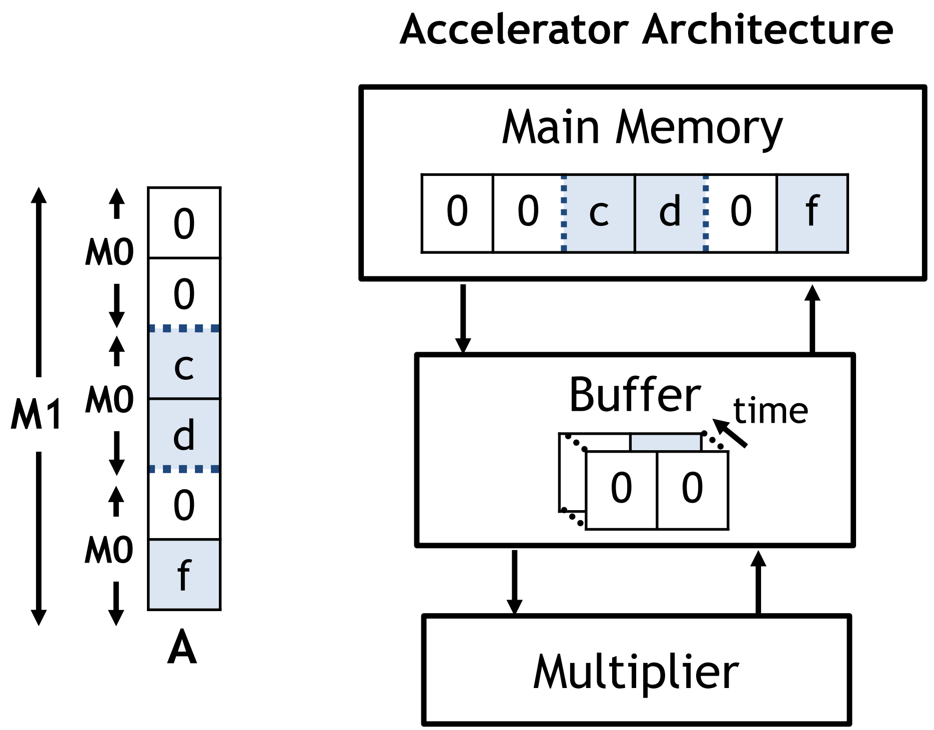


The format-agnostic nature of fibertree allows clean separation of the sparse nature of tensor and its format

Decides the theoretical savings sparse optimization features can bring

One of the implementation decisions to realize sparse optimization features

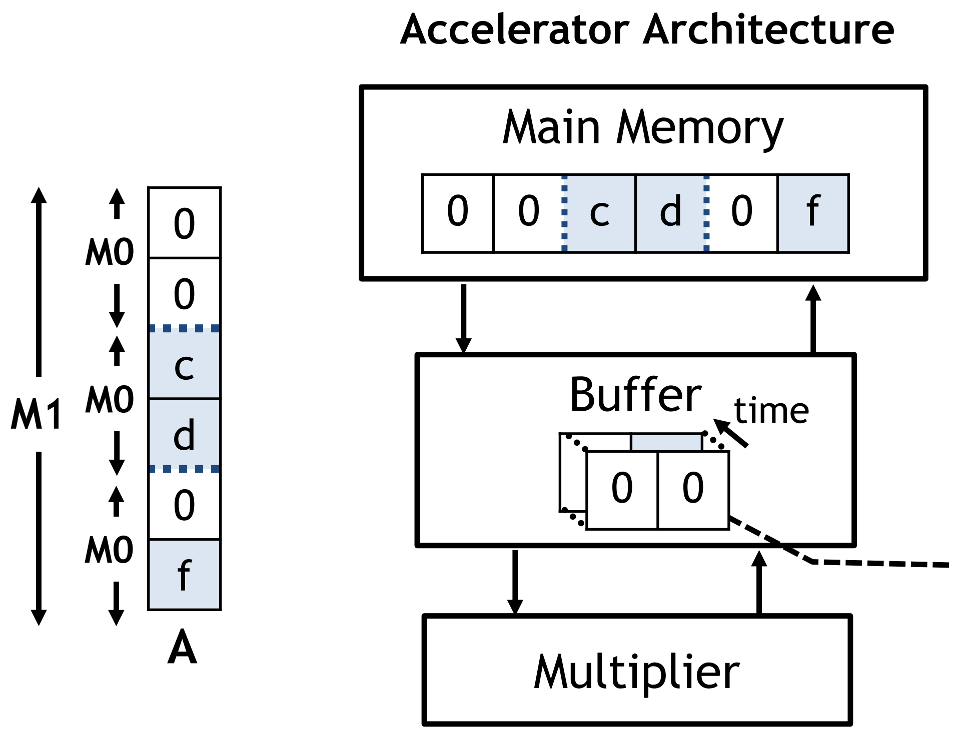
Mapping Introduces Tiled Tensors



Mapping

```
----- Main Memory -------
for m in [0:M1)
----- Buffer -------
  for m in [0:M0)
```

Mapping Introduces Tiled Tensors



Mapping

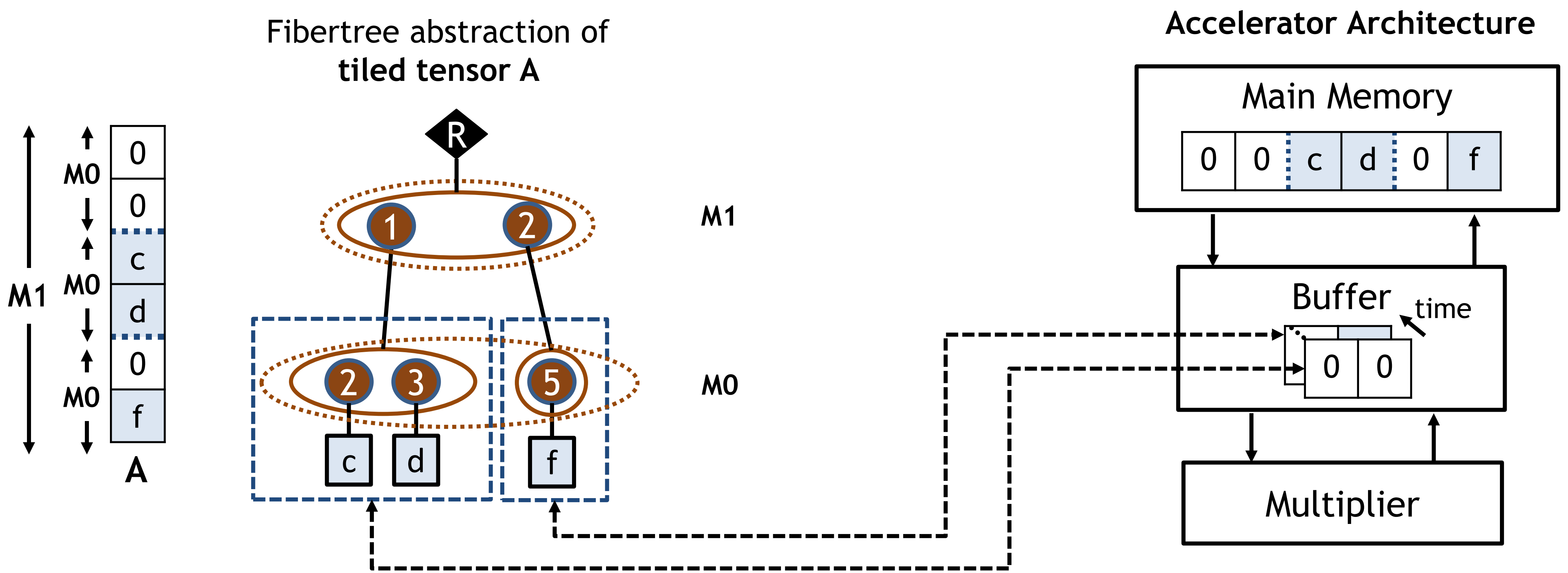
```
----- Main Memory -------
for m in [0:M1)
----- Buffer -------
  for m in [0:M0)
```

Final questions to answer

- How much capacity is needed to store the subtile?
- How much data transfers are there between storages?
- …

All dependent on the sparse nature of the (sub)tensor, i.e., how many nonzeros values in (sub)tensor

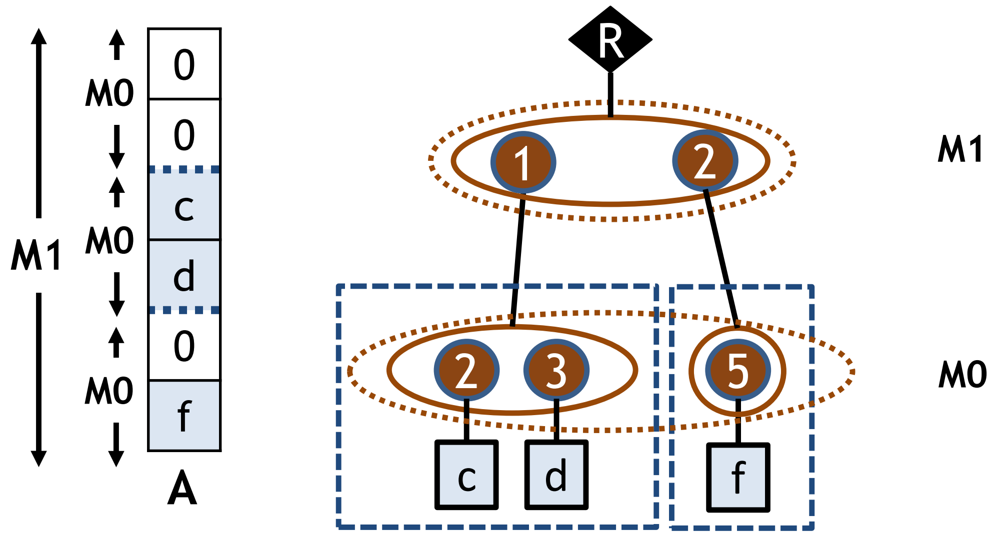
Fibertree Defines the Sparse Nature of Tensors



Characterizing the sparse nature of a (sub)tensor
==
Characterizing a fiber

Fibertree Defines the Sparse Nature of Tensors

Fibertree abstraction of
tiled tensor A



To characterize all the fibers in the tensor, we need to consider

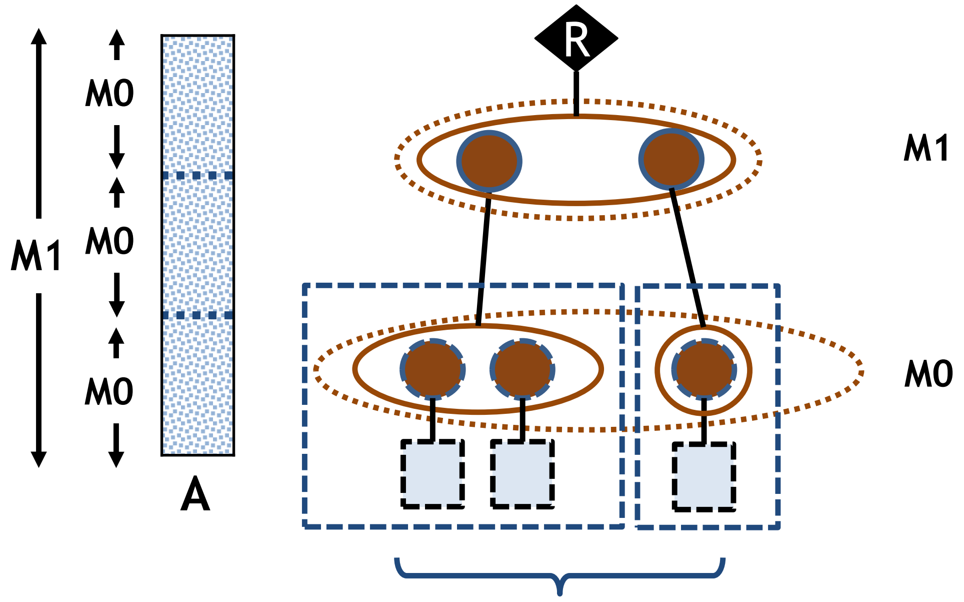
- # of ranks
- # of fibers in each rank
- # of elements in each fiber, i.e., fiber occupancy

Deterministic when exact data can be examined

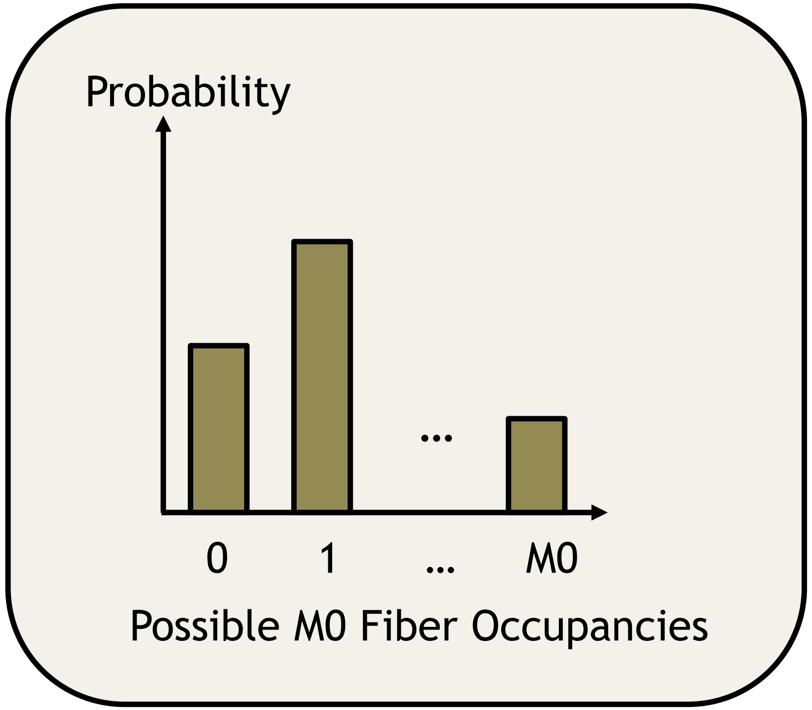
Statistical Density Models Necessary for Analytical Modeling

To ensure fast modeling speed, analytical modeling cannot examine the exact data in fibers

Non-deterministic fibertree abstraction of tiled tensor A



Without exact data, the **# of fibers** and **# of elements in each fiber** cannot be determined

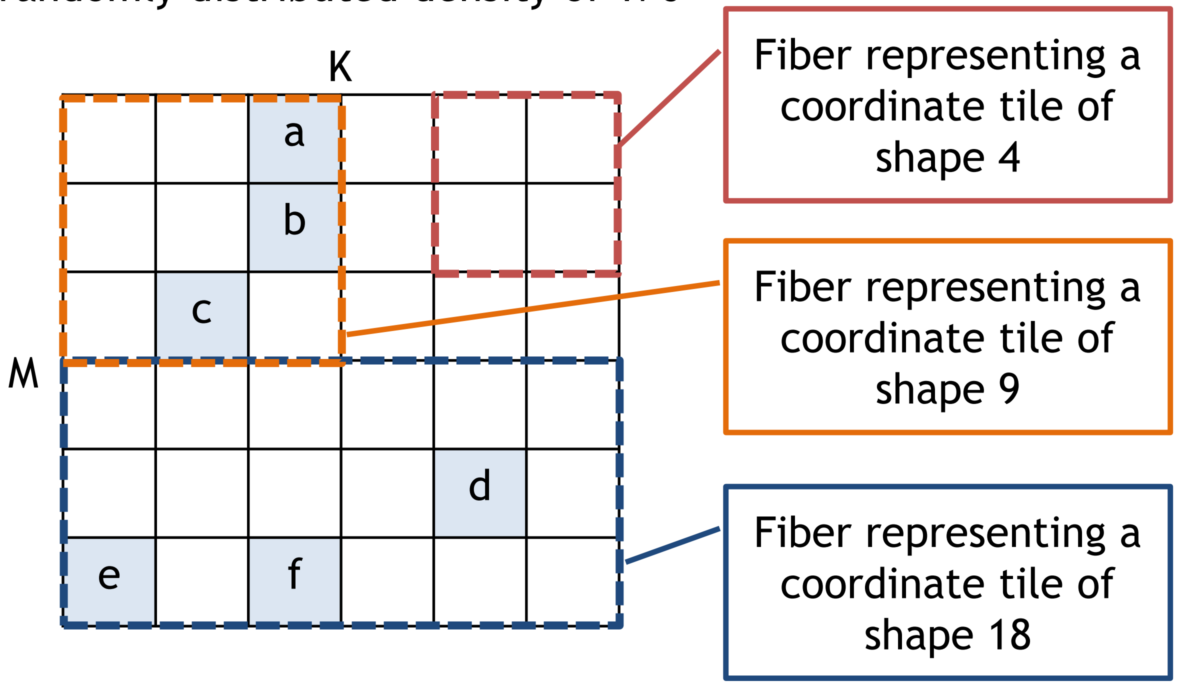


Probability distributions depend on the choice of statistical workload density model

Density Model 1: Hypergeometric Distribution

Describes the randomly distributed zeros in a tensor

Example 6x6 tensor with
randomly distributed density of 1/6

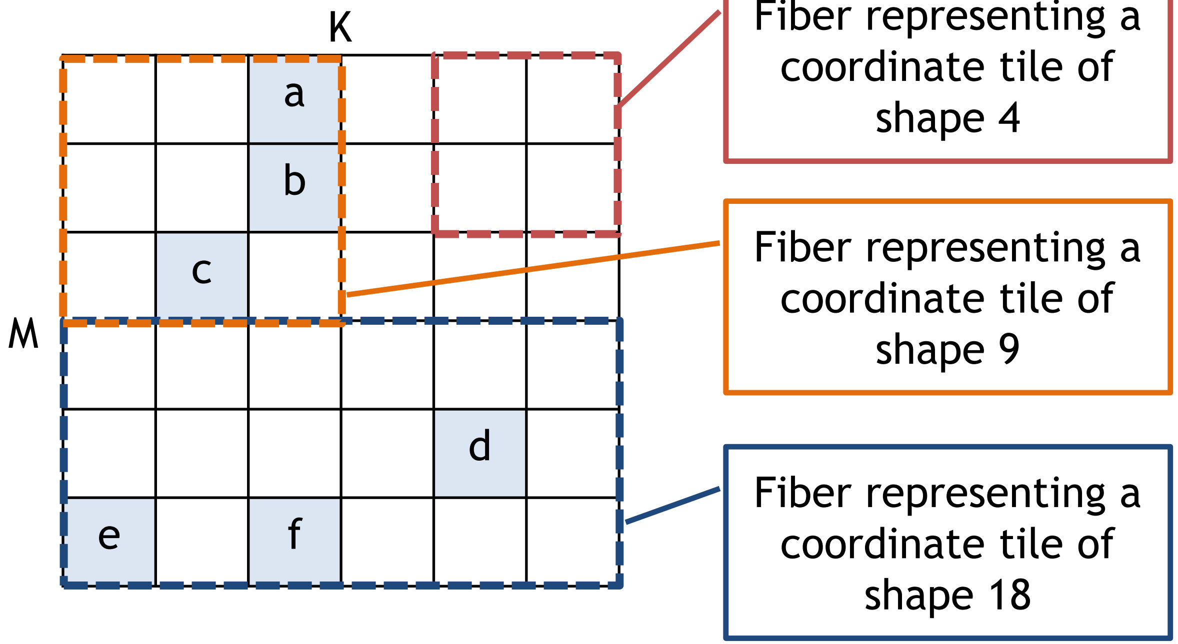


Main Characteristics

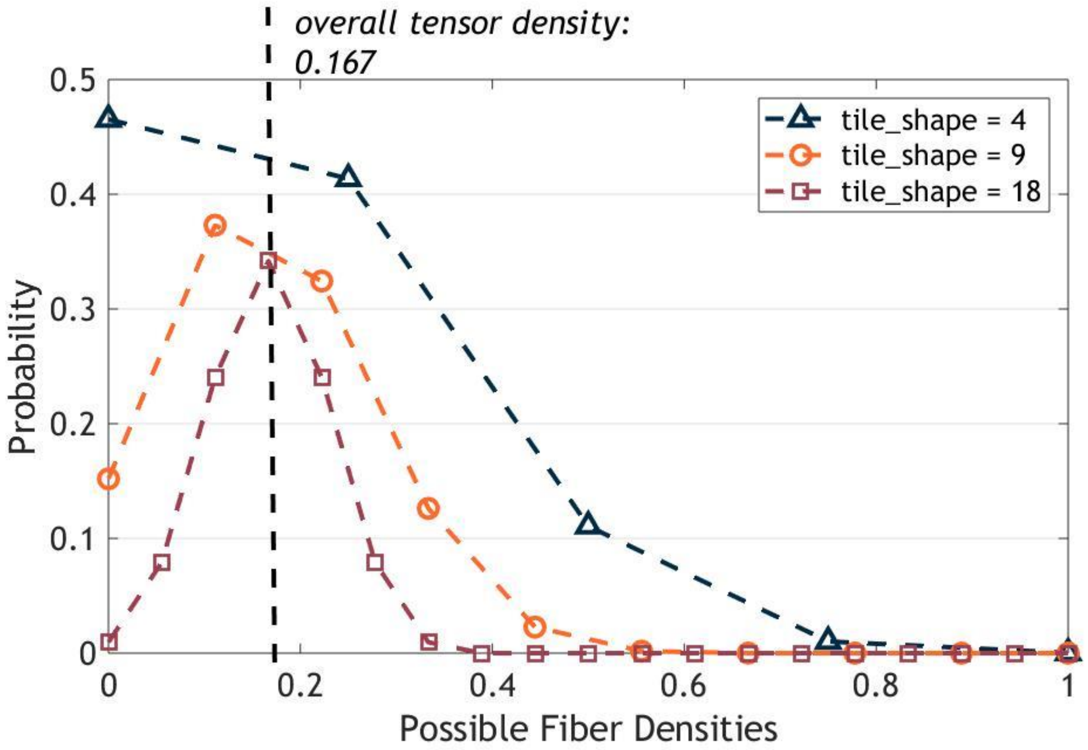
The smaller the tile is, the more likely for the fiber to be empty/full (low density/high density)

Density Model 1: Hypergeometric Distribution

Example 6x6 tensor with
randomly distributed density of 1/6



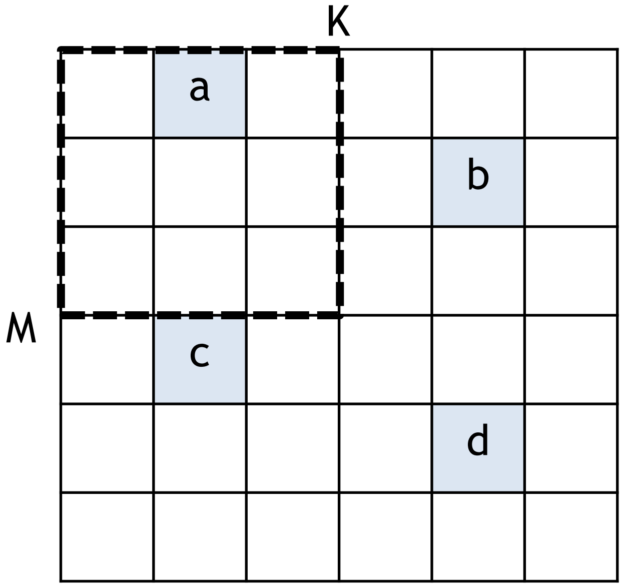
Fiber Densities Characterized By
Hypergeometric Model



Density Model 2: Fixed-Structured Distribution

Describes a structured distribution of zeros in a tensor, where all tiles in the tensor have a shared fixed density

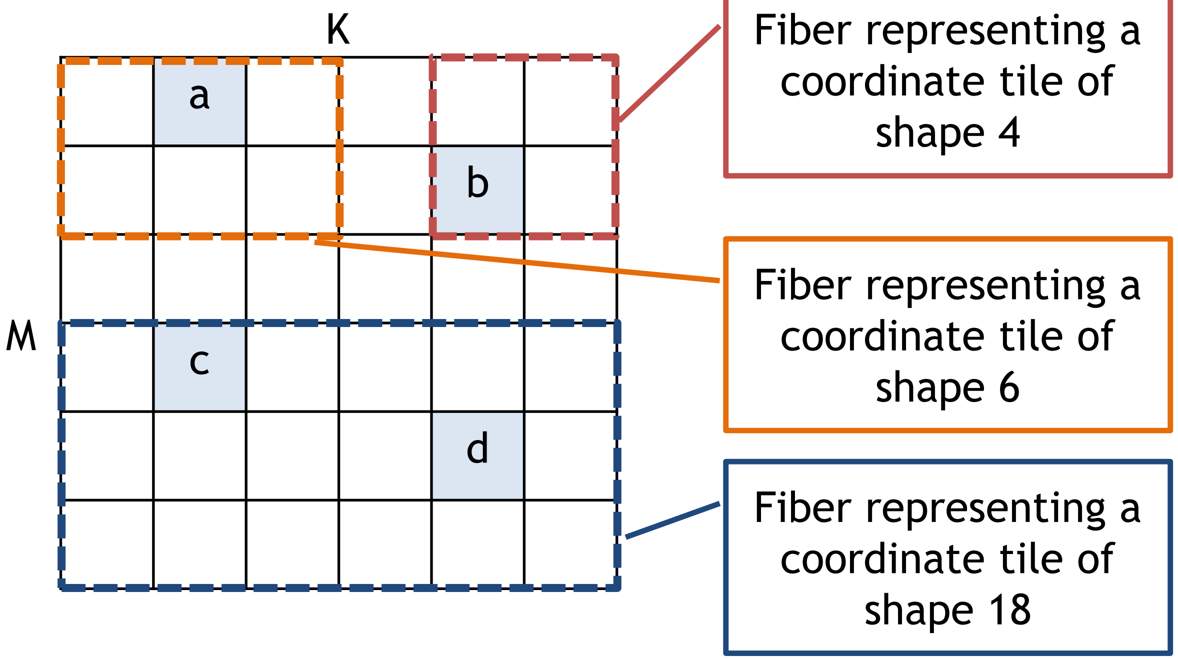
Example 6x6 tensor with
a fixed structured density of 1/9



Density Model 2: Fixed-Structured Distribution

Describes a structured distribution of zeros in a tensor, where all tiles in the tensor have a shared fixed density

Example 6x6 tensor with
a fixed structured density of 1/9



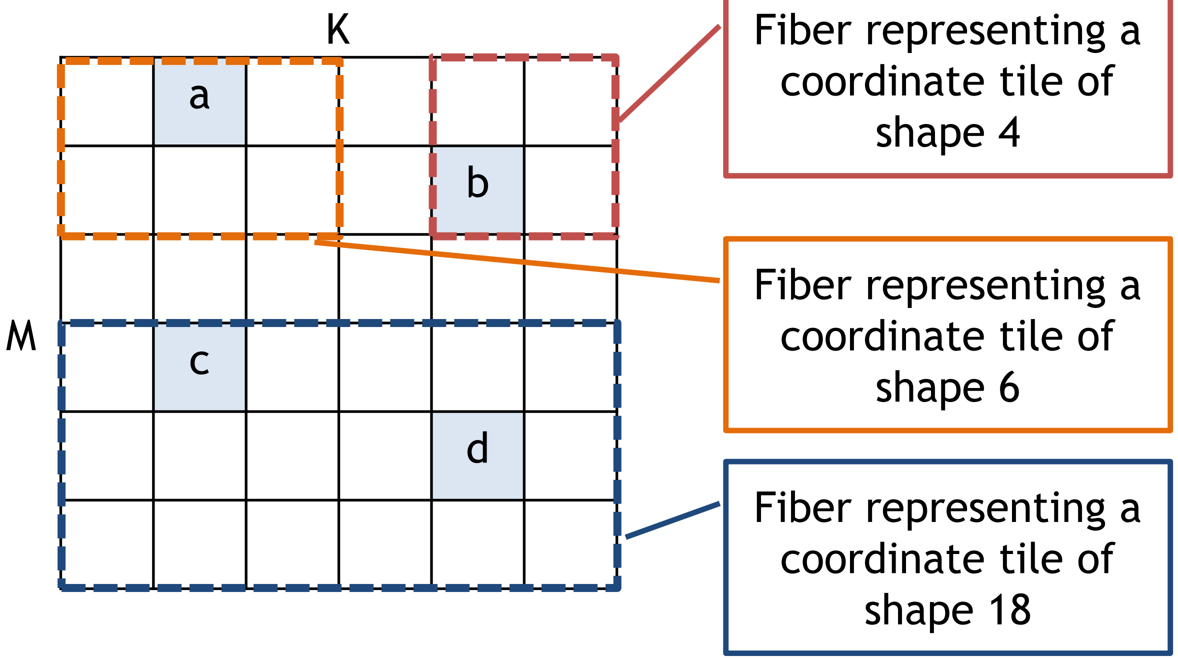
Main Characteristics

Fibers might have non-deterministic occupancy if tile *shape x fixed density* is non-integer

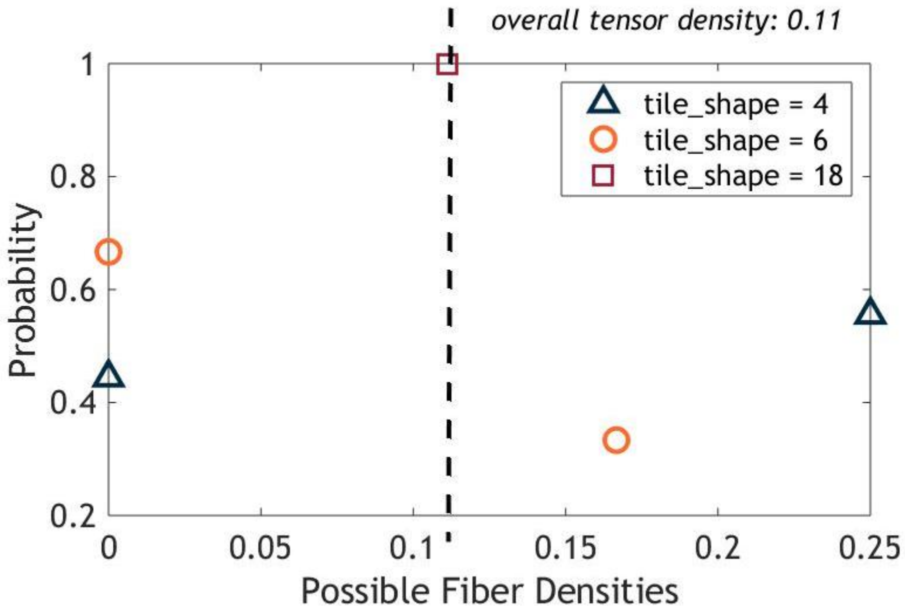
Density Model 2: Fixed-Structured Distribution

Non-integer occupancy represented as weighted sum of integer possible occupancies

Example 6x6 tensor with
a fixed structured density of 1/9



Fiber Densities Characterized By
Fixed-Structured Density Model



Specifications and Their Interactions

Mapping

Statistical Workload Density Models

Sparse Optimization Features

Interactions

Statistical characterization of fiber occupancies

- Required hardware capacity at each level
- Resulting sparse traffic of the storage and compute units

Proposed Sparse Tensor Accelerator Modeling Methodology

Sparse Optimization Feature Impact Modeling

Specifications and Their Interactions

Mapping

Statistical Workload Density Models

Sparse Optimization Features

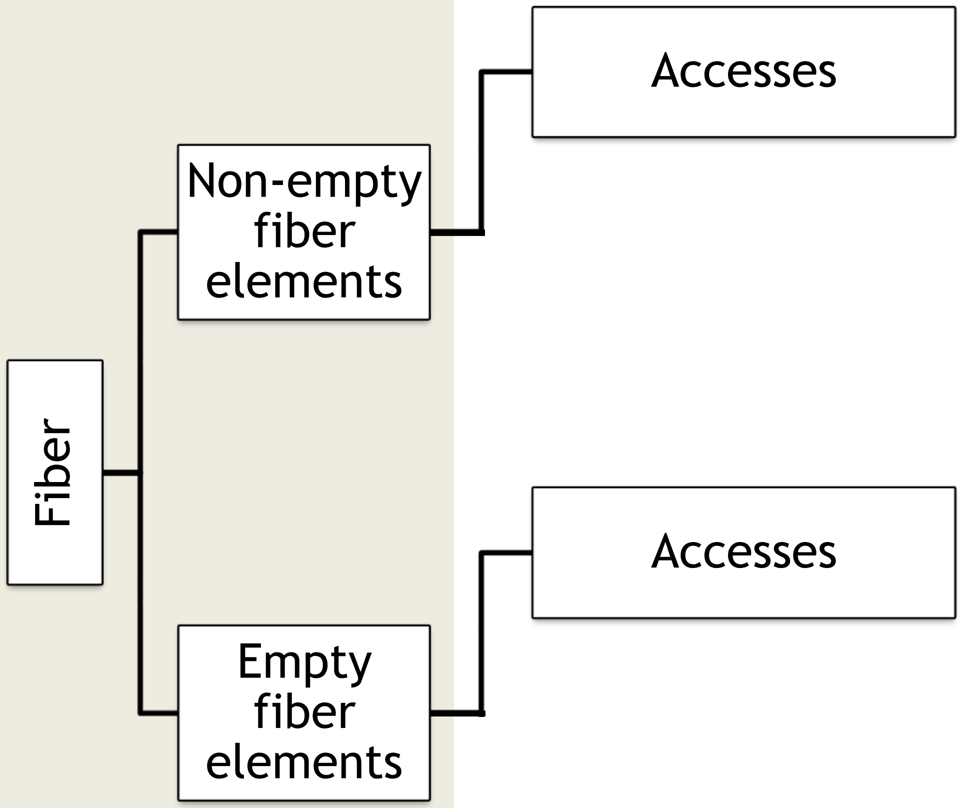
Interactions

Statistical characterization of fiber occupancies

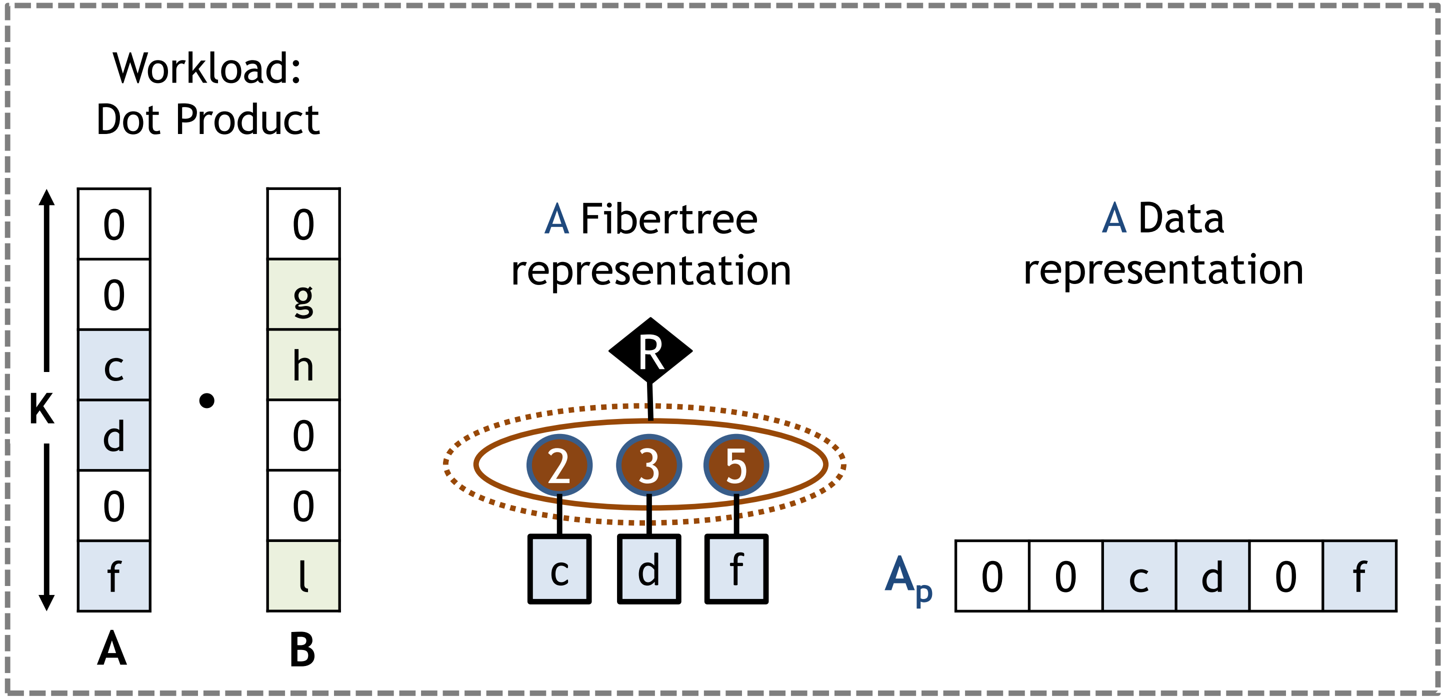
- Required hardware capacity at each level
- Resulting sparse traffic of the storage and compute units

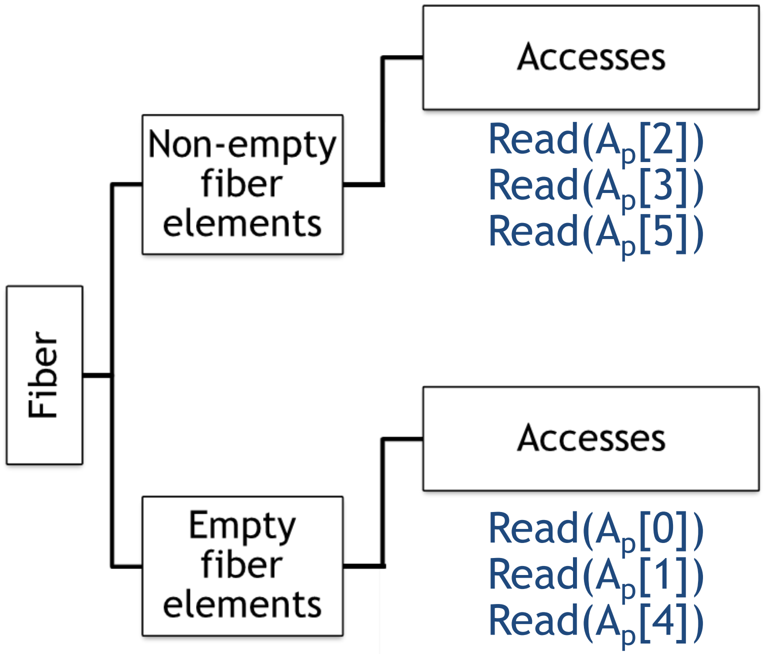
Baseline Storage Access Types Related to a Fiber

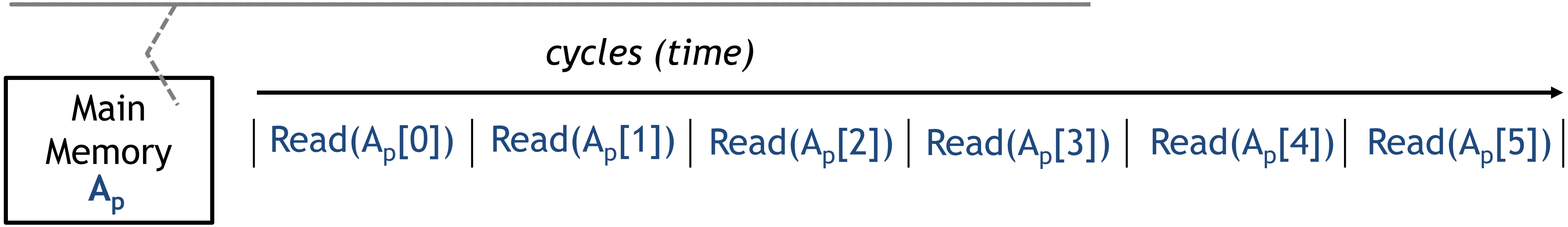
Deterministic based on the statistical occupancy of fiber



Baseline A Tensor Accesses in A Dot Product Workload

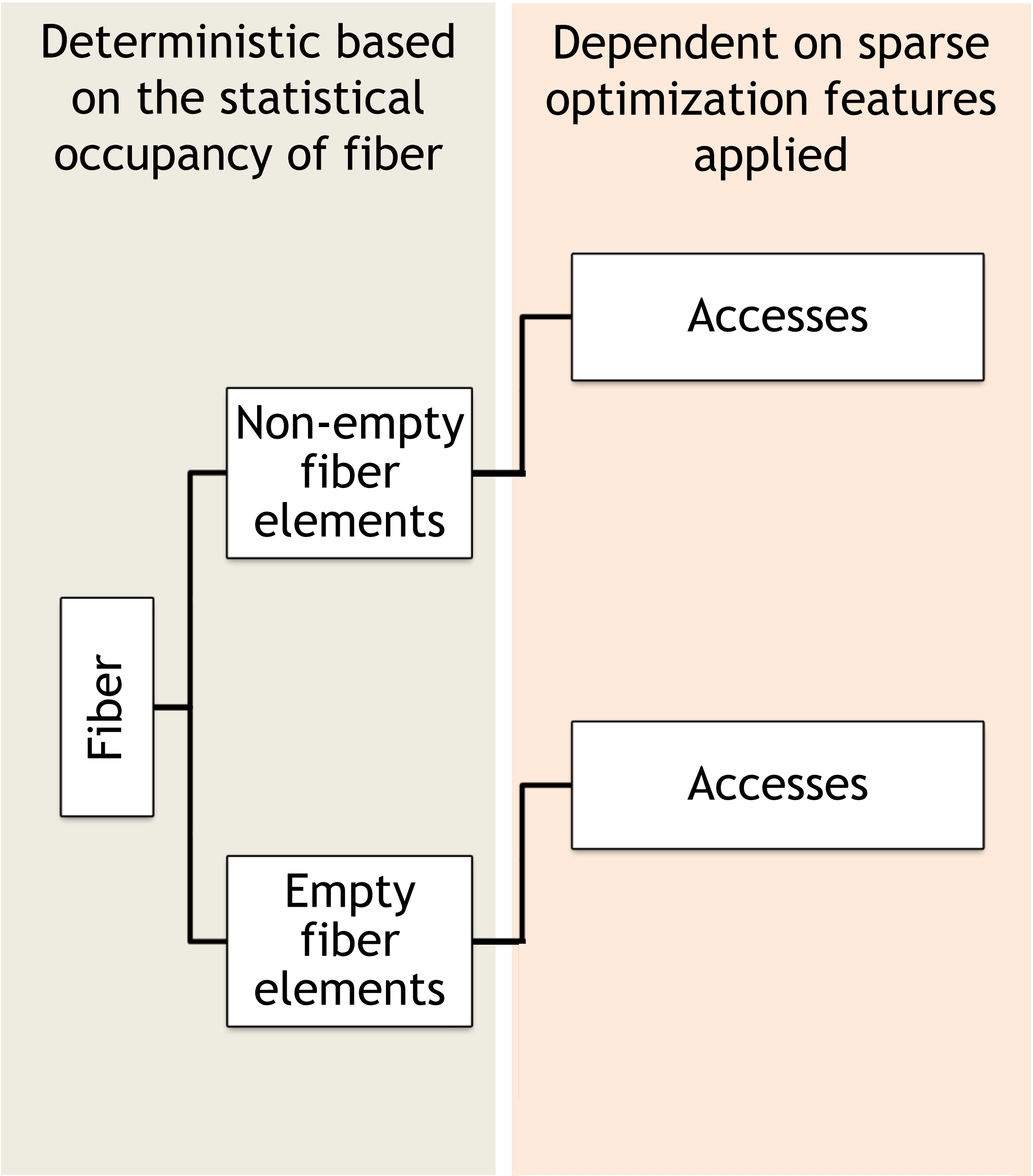




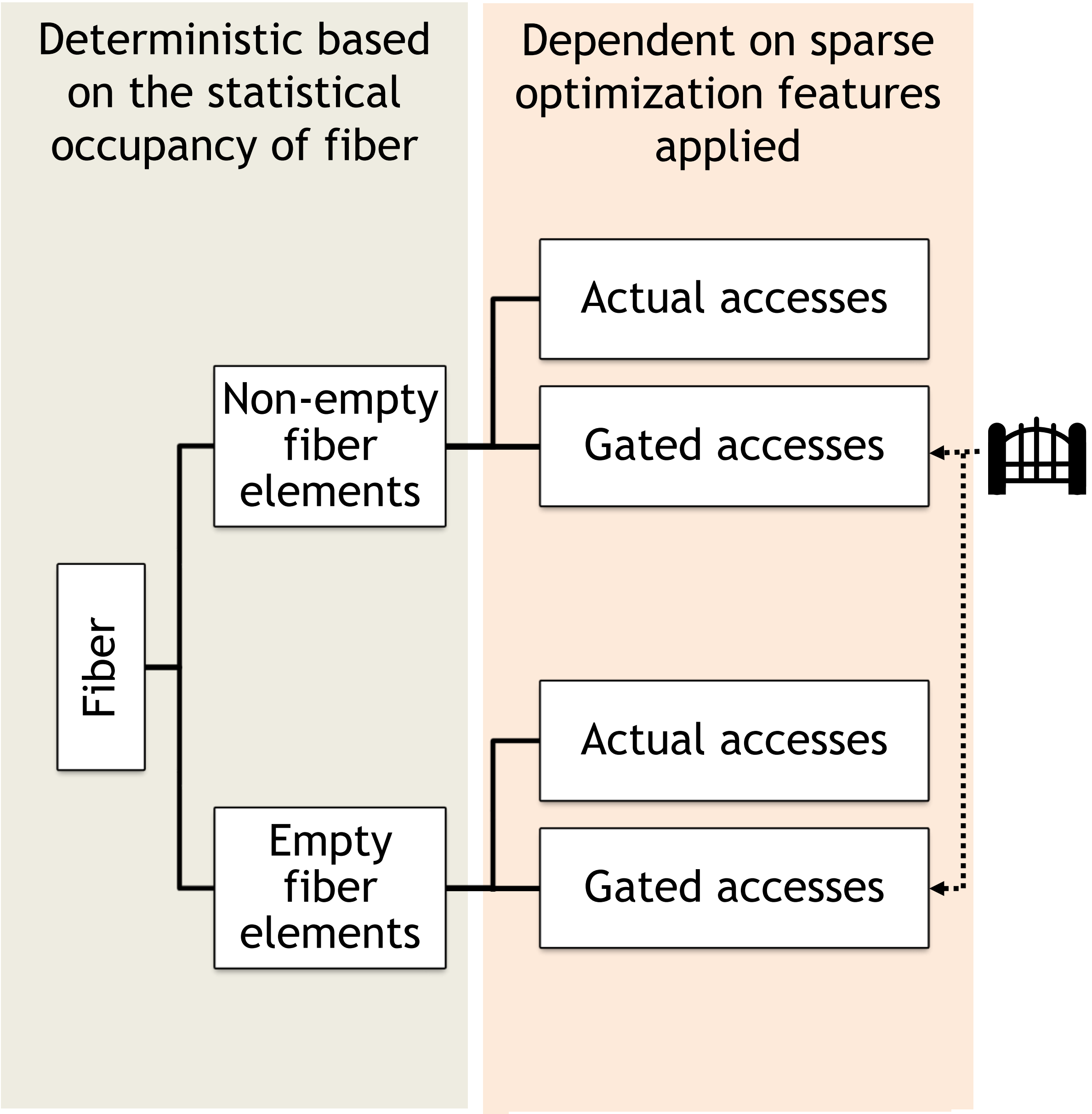


Total: 6 actual accesses, 6 cycles

Sparse Optimization Features Reduces Actual Accesses



Gating Leads to Gated Accesses

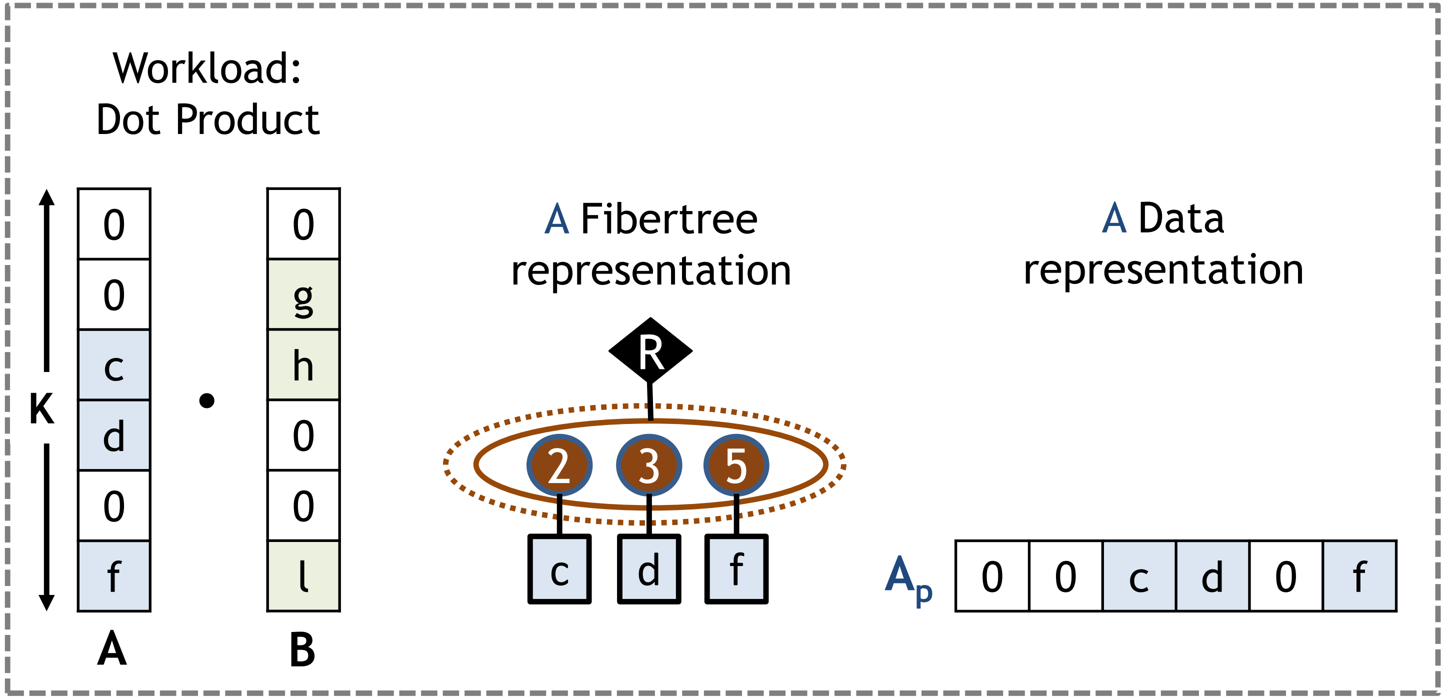


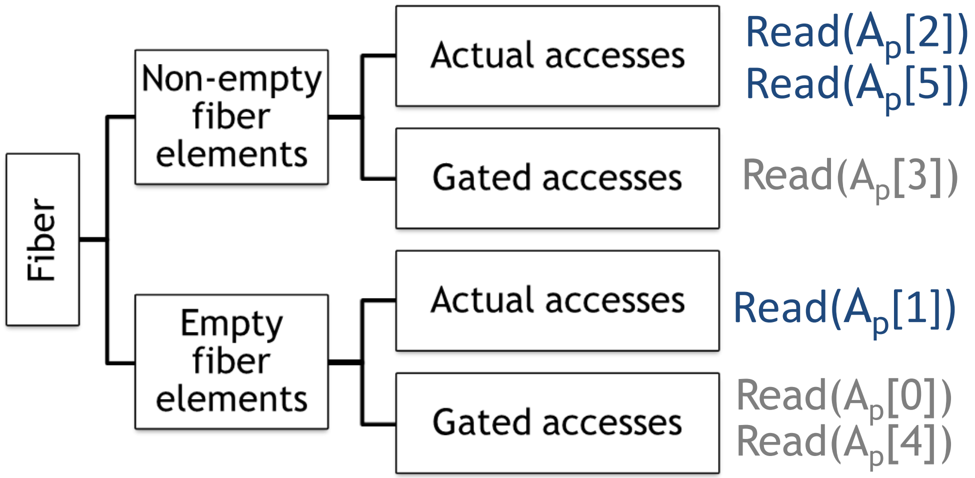
Gating:

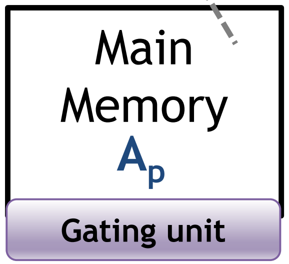
Explicit energy saving of access to the payload* of one element of a fiber based on the emptiness of an element of another fiber

**Note that since the "payload" of an element of a fiber may be a whole fiber (or tree of fibers) more than one accesses can be optimized*

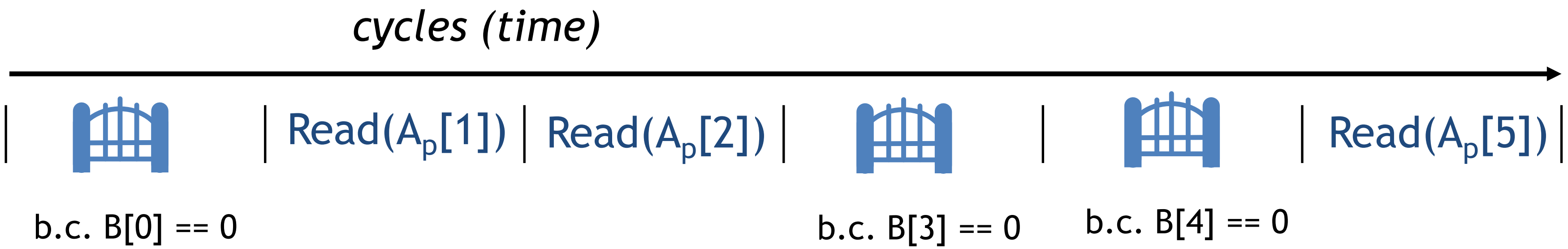
Zero-Gated A Tensor Accesses in A Dot Product Workload





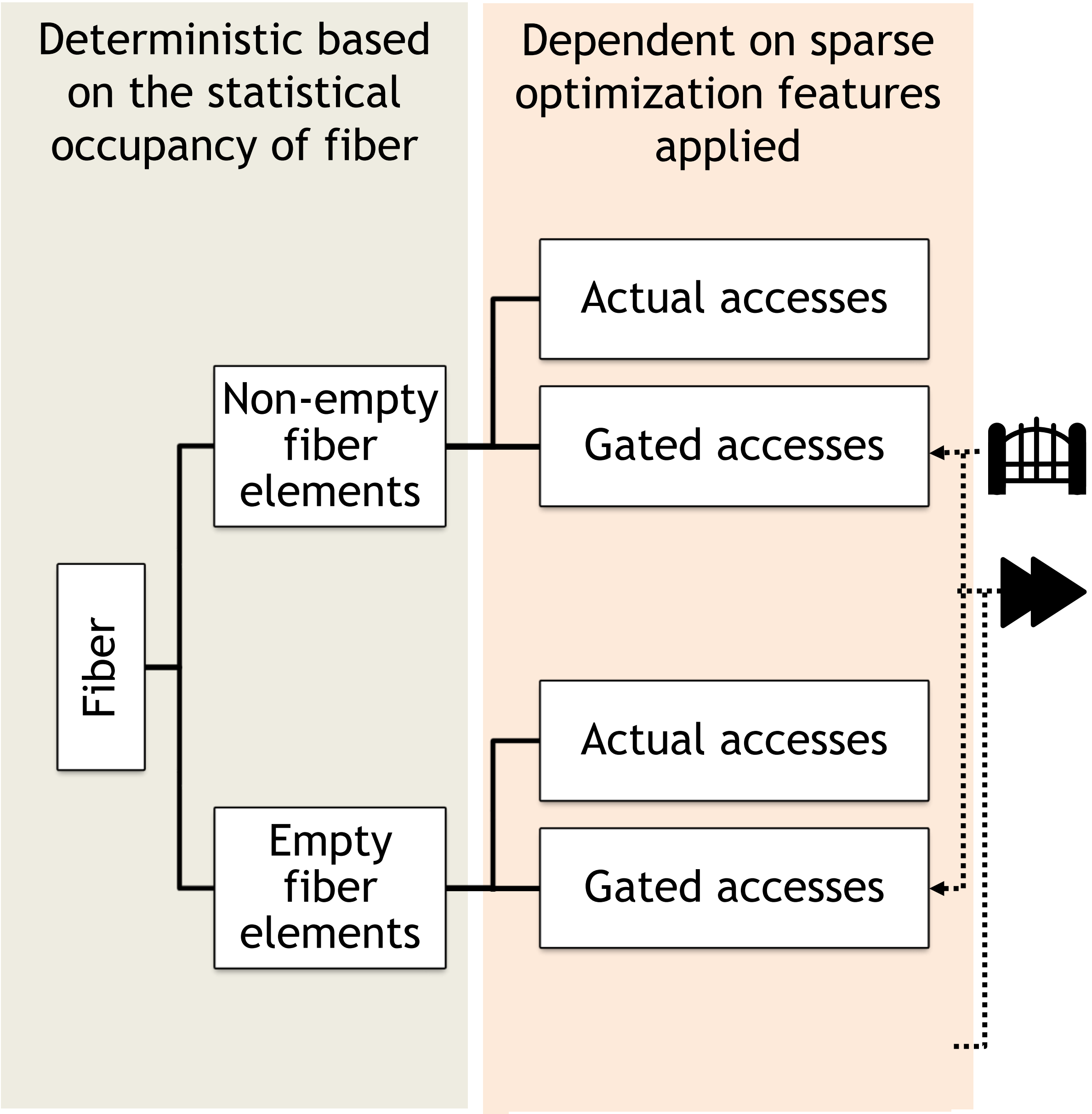


Gate A based on B



Total: 3 actual accesses, 6 cycles

Skipping Leads to Skipped Accesses

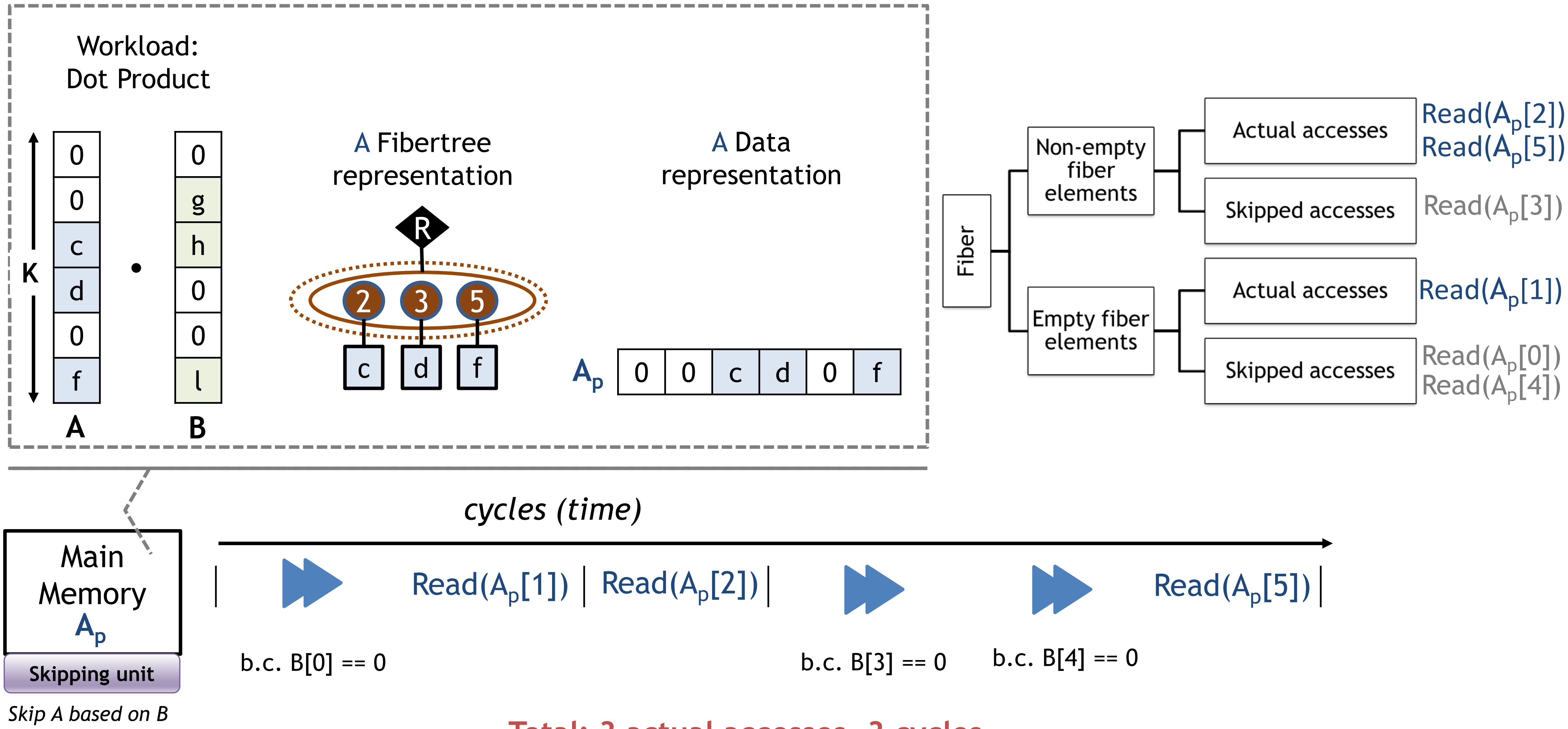


Gating:
Explicit energy saving of access to the payload* of one element of a fiber based on the emptiness of an element of another fiber

Skipping:
Explicit skipping over access to the payload* of one element of a fiber based on the emptiness of an element of another fiber

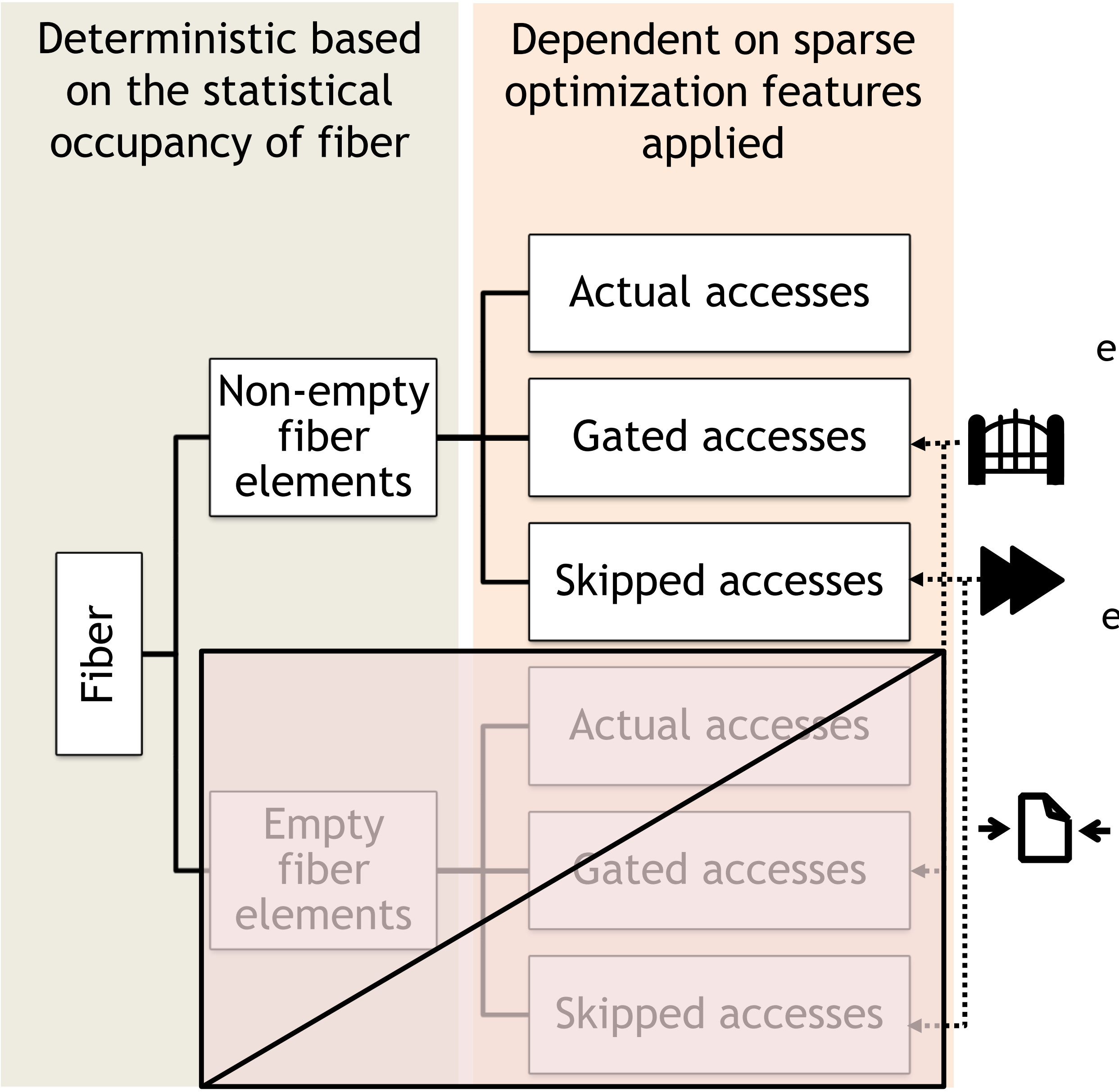
**Note that since the "payload" of an element of a fiber may be a whole fiber (or tree of fibers) more than one accesses can be optimized*

Zero-Skipped A Tensor Accesses in A Dot Product Workload



Total: 3 actual accesses, 3 cycles

Compression Eliminates Accesses to Empty Elements



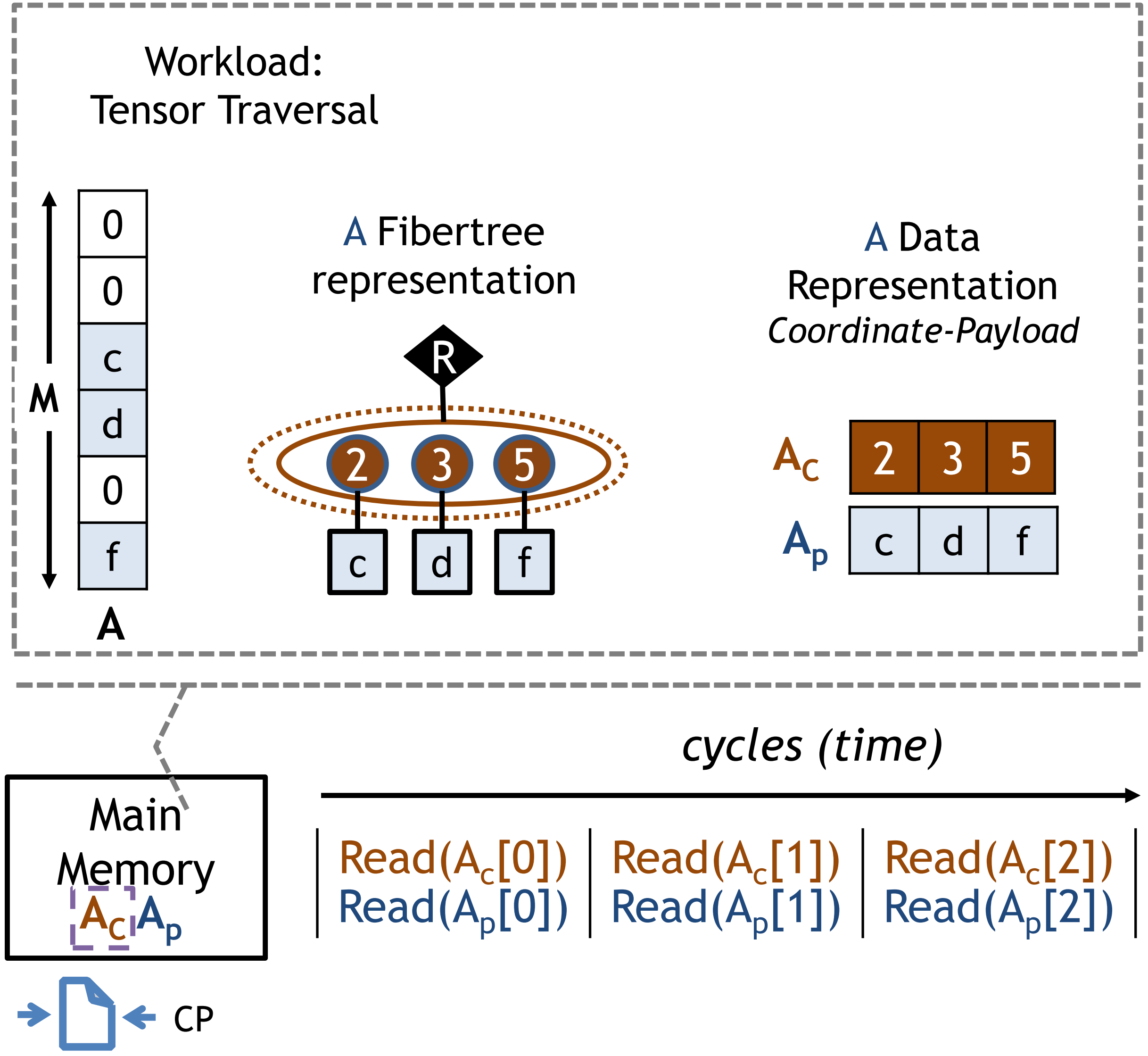
Gating:
Explicit energy saving of access to the payload* of one element of a fiber based on the emptiness of an element of **another fiber**

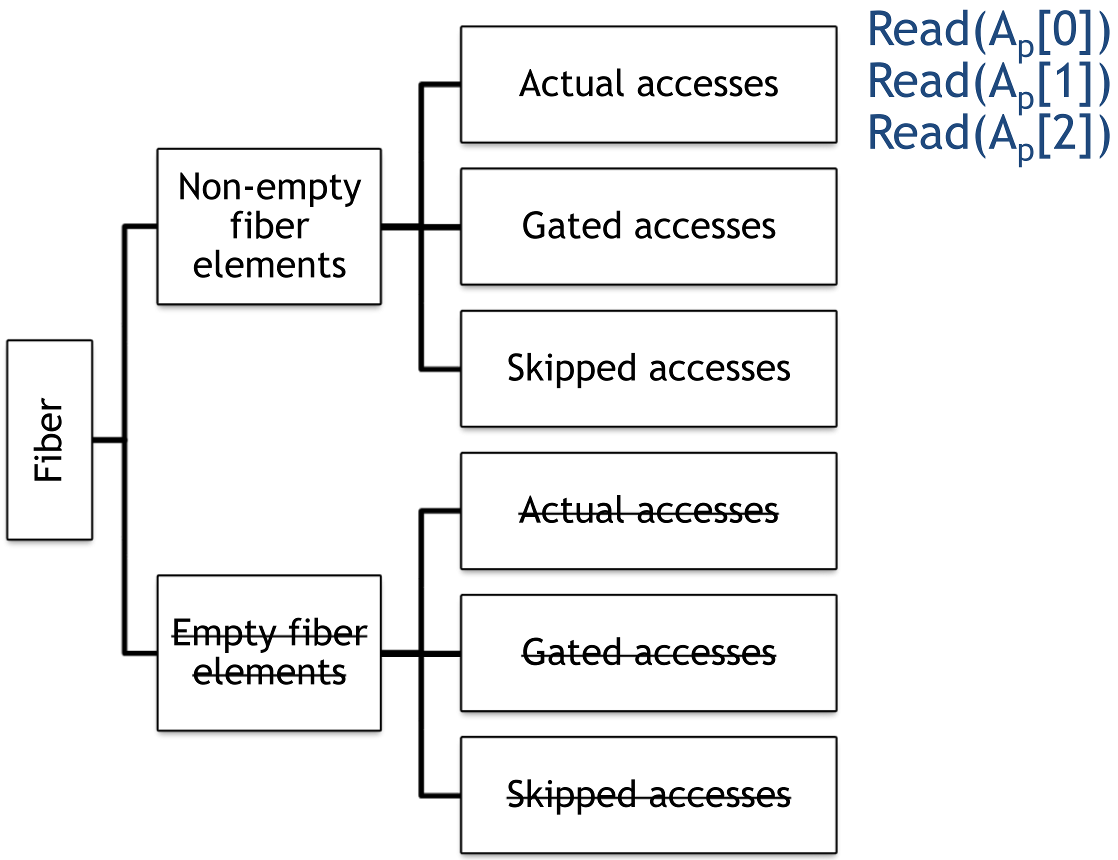
Skipping:
Explicit skipping over access to the payload* of one element of a fiber based on the emptiness of an element of **another fiber**

Format:
Choose data representation formats to save storage space and/or allow better realization of gating and skipping

**Note that since the "payload" of an element of a fiber may be a whole fiber (or tree of fibers) more than one accesses can be optimized*

A Tensor Traversal with Coordinate Payload Format

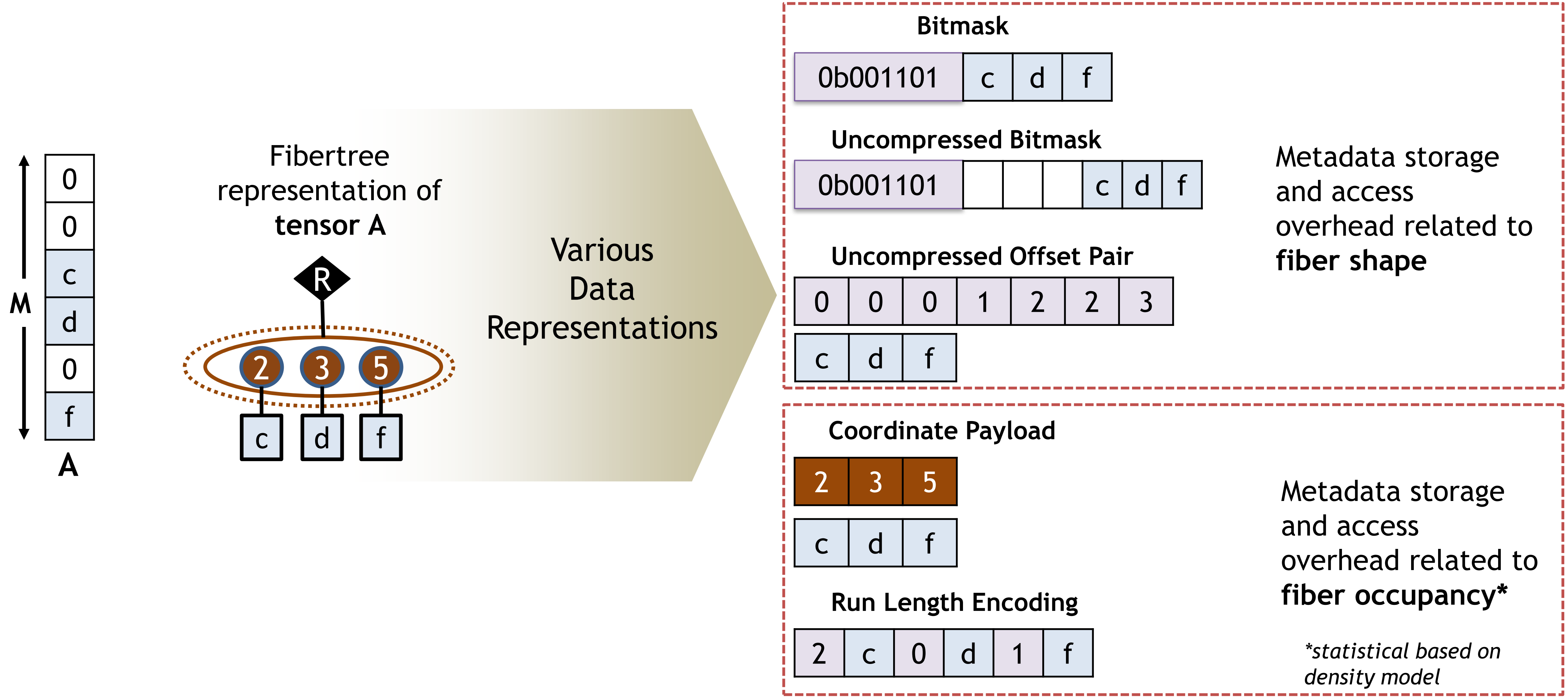




Total: 3 actual data accesses, 3 cycles, 3 coordinate metadata accesses

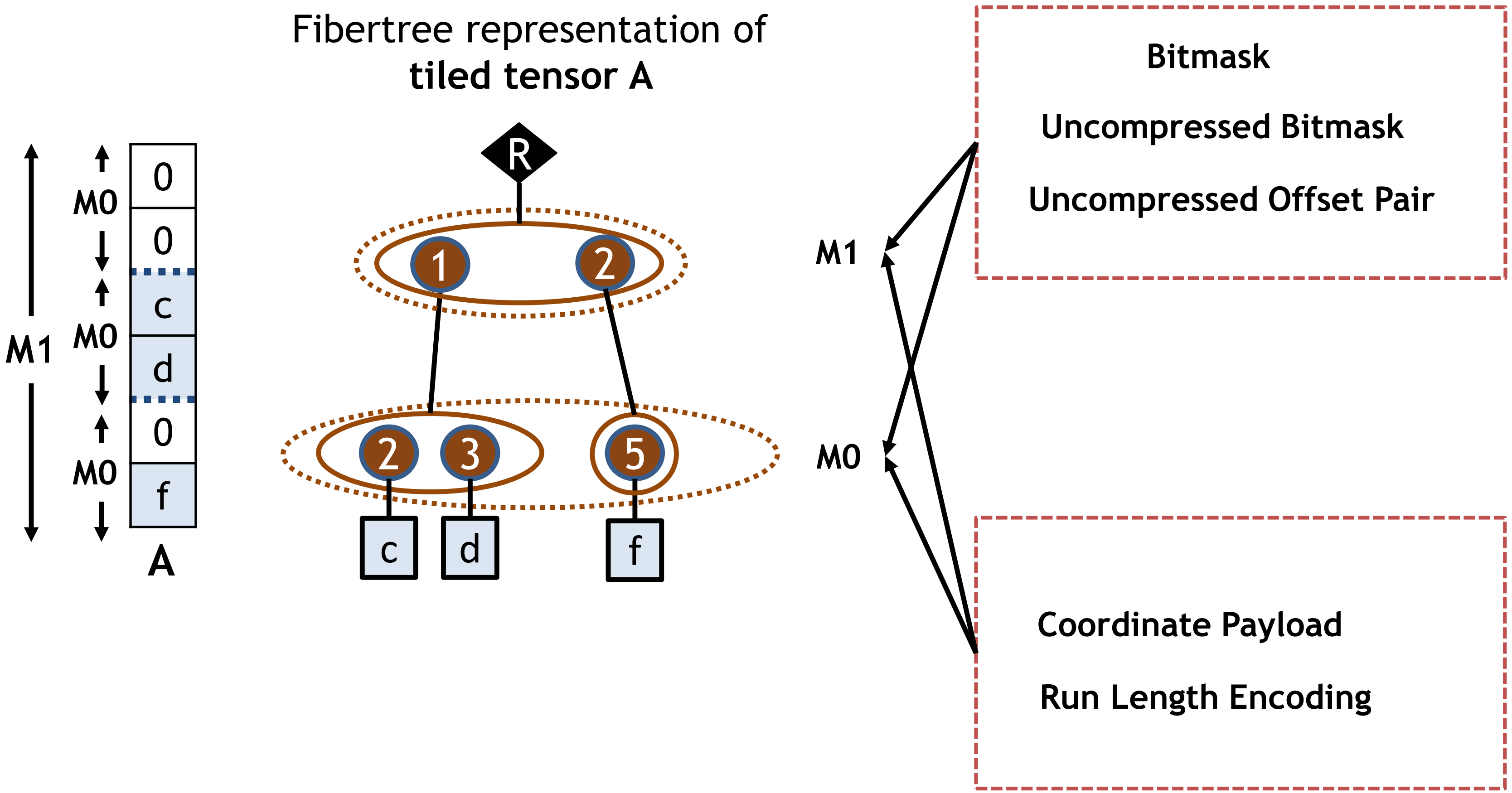
Format Choice Leads to Metadata Overhead

Metadata that identifies the locations of zeros is necessary



Multi-Rank Metadata Overhead

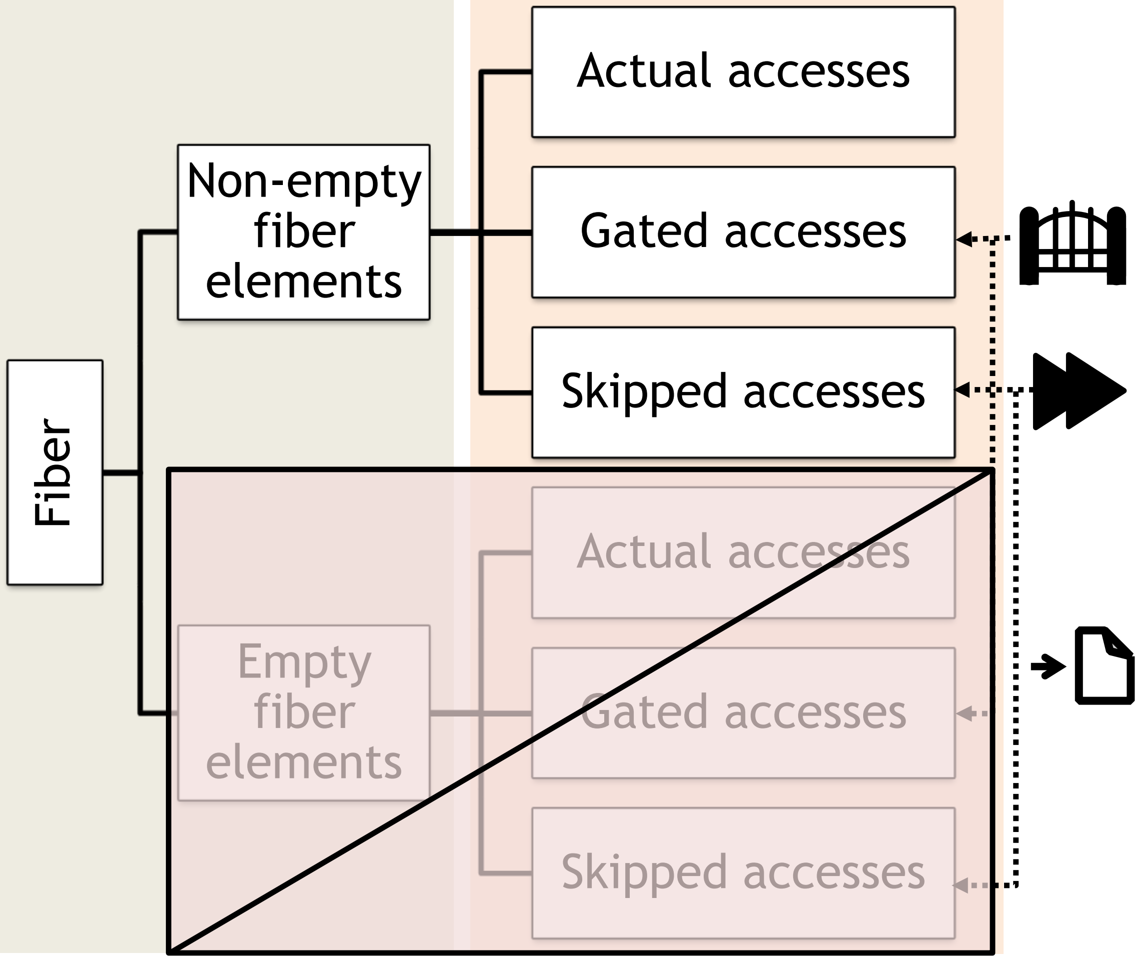
Per-Rank Occupancy and Access Analysis Allows Modeling of Arbitrary Compression Format



Impact Defined by Fibers in Different Tensors

Deterministic based on the statistical occupancy of fiber

Dependent on sparse optimization features applied



Gating:
Explicit energy saving of access to the payload* of one element of a fiber based on the emptiness of an element of another fiber

Skipping:
Explicit skipping over access to the payload* of one element of a fiber based on the emptiness of an element of another fiber

Dependent on another tensor's density

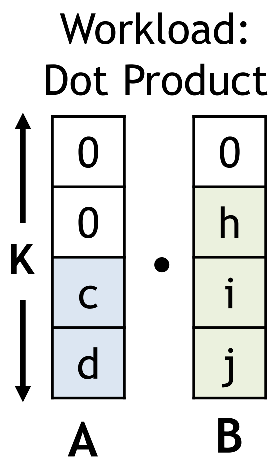
Format:
Choose data representation formats to save storage space and/or allow better realization of gating and skipping

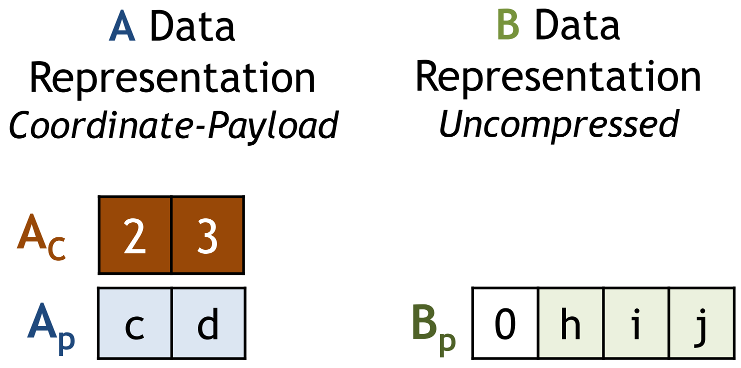
Dependent on the tensor's own density

**Note that since the "payload" of an element of a fiber may be a whole fiber (or tree of fibers) more than one accesses can be optimized*

Interplay Between Different Sparse Optimization Features

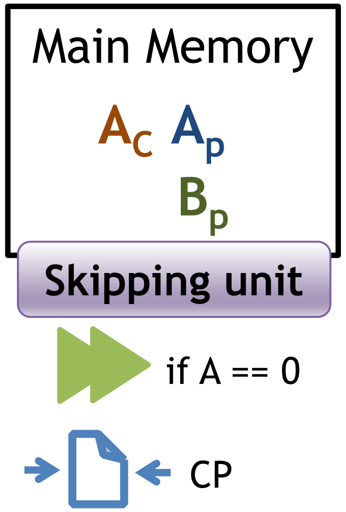
Multiple sparse optimization features can be applied at the same time
As a result, the impact on required storage capacity and storage accesses aggregates

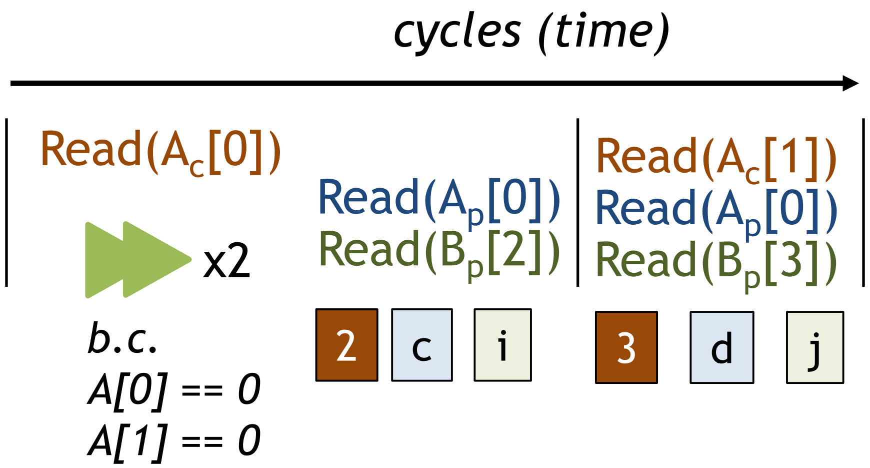





```
for (a_c, a_p) in A:
 Z[a_c] += a_p * B_p[a_c]
```

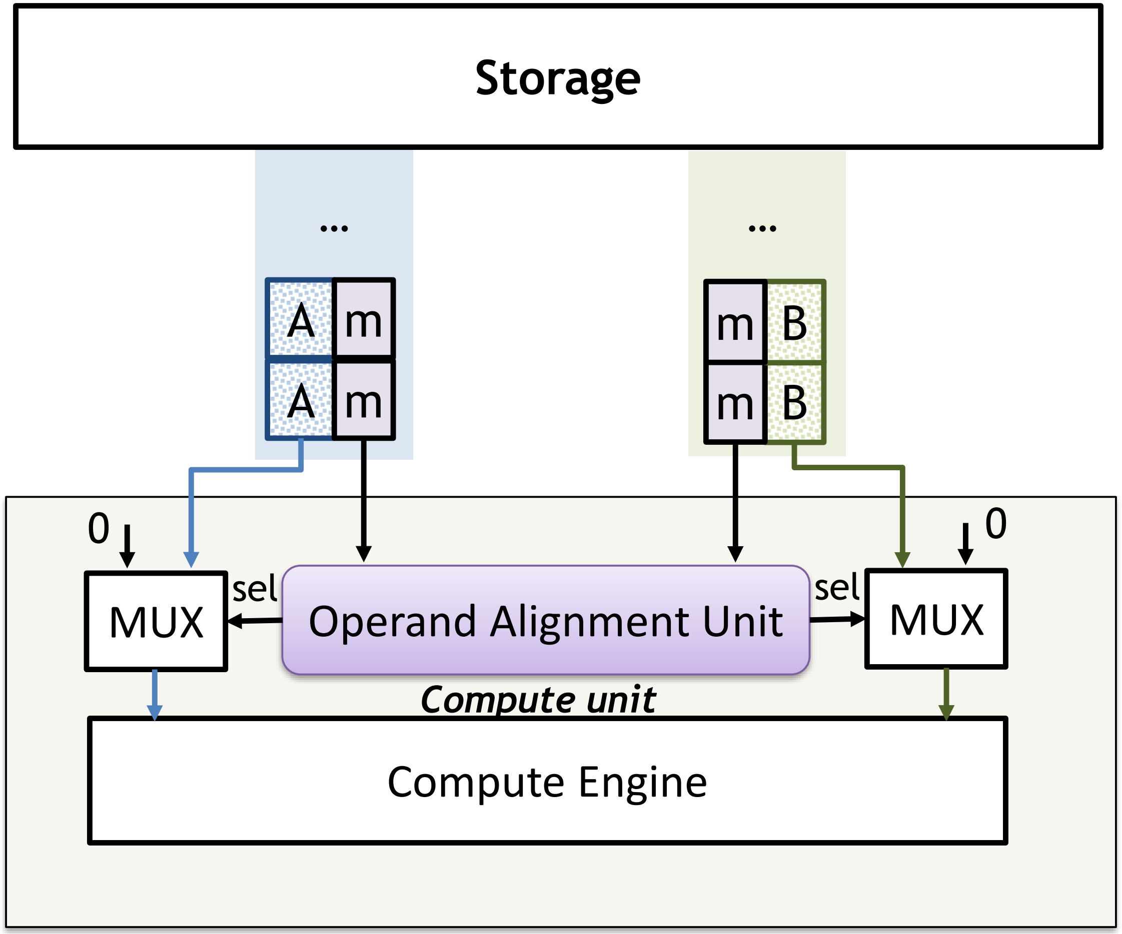
Each A_c value is 2 bits
Each A_p value is 8 bits
Each B_p value is 8 bits





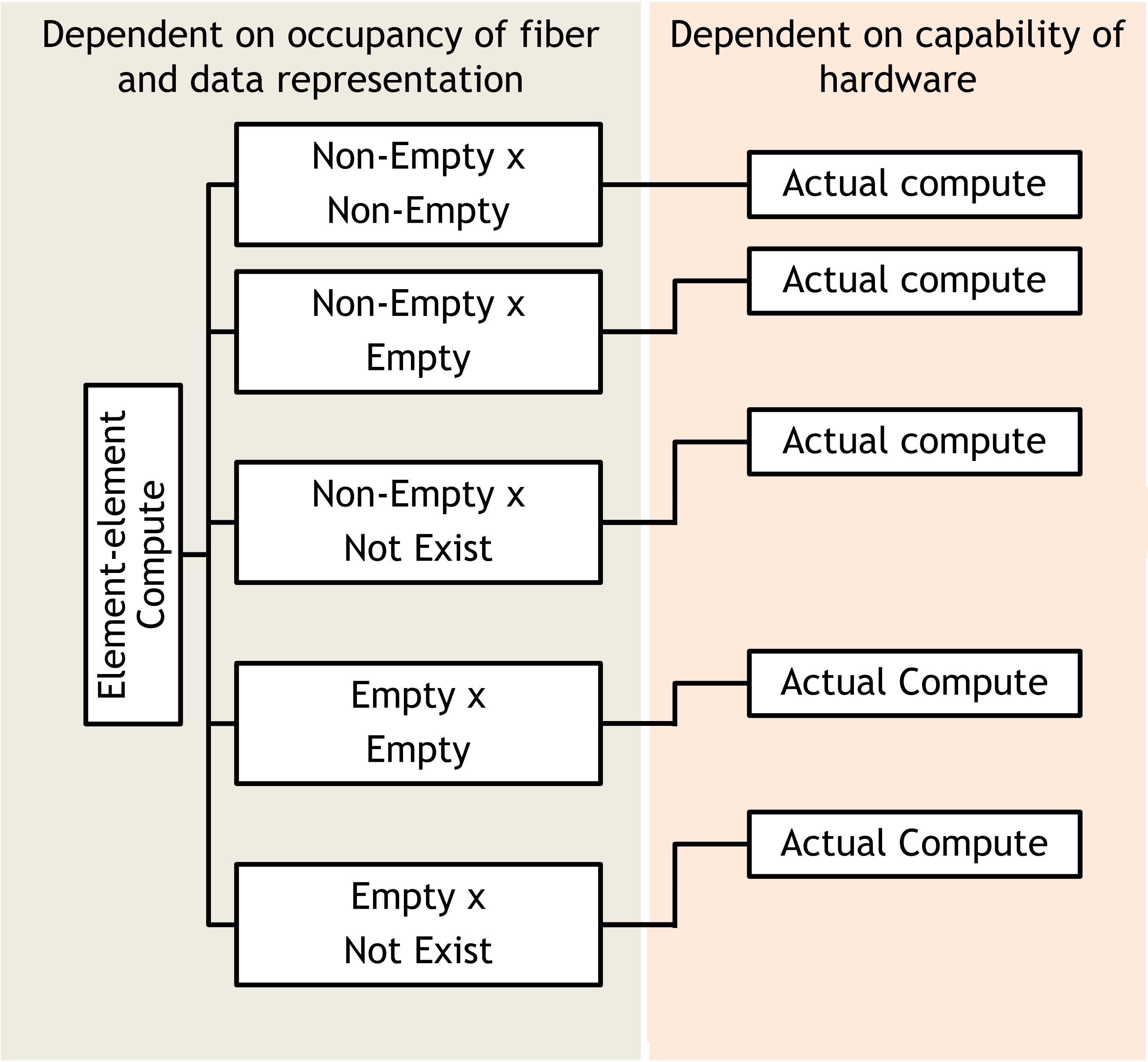
- Processing time reduced by 2x
- Hardware capacity requirement reduced by 1.23x
- Number of payload storage accesses reduced by 2x
- Incurs 2 extra metadata storage access overhead

Baseline Compute Unit Hardware Setup

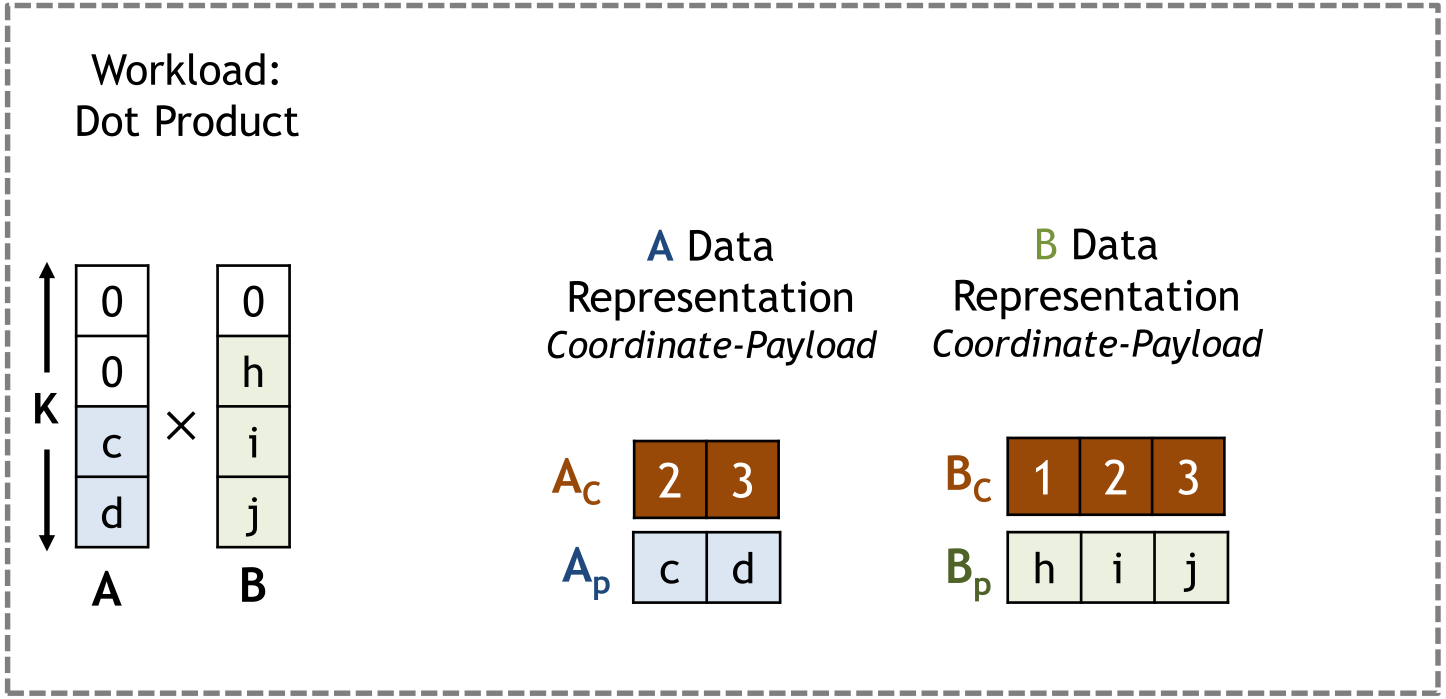


Operand alignment unit checks operand metadata and decides whether the incoming operands correspond to each other

Sparse Optimization Features Lead to Different Types of Computes

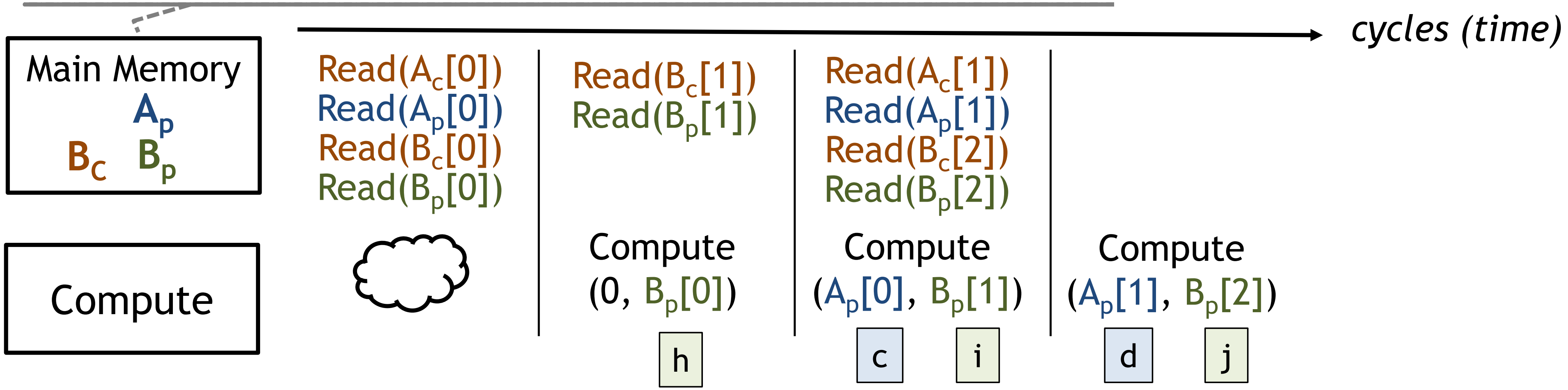


Baseline Compute Unit Working on Dot Product

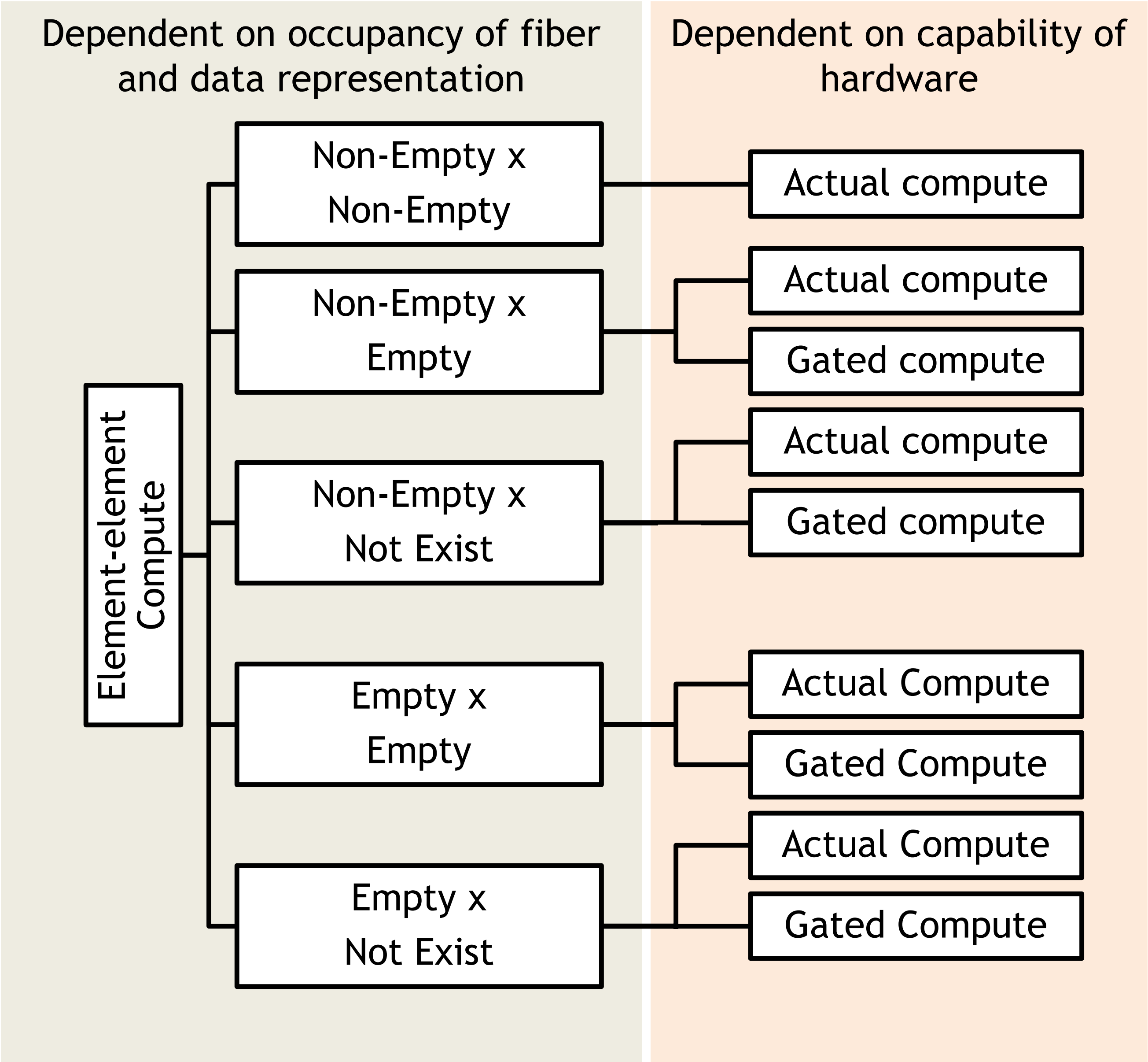


K: contracted dimension

It is important to align the contracted dimension to perform a valid compute



Sparse Optimization Features Lead to Different Types of Computes



Gating:
Explicit energy saving of compute when one of the payloads of operand elements is empty (i.e., compute engine recognizing zero operands)

Gated Compute Unit Working on Dot Product

Workload:
Dot Product

K	A		B
	0		0
	0		h
	c	×	i
	d		j

A **B**

A Data Representation
Coordinate-Payload

A_C	2	3
A_p	c	d

B Data Representation
Coordinate-Payload

B_C	1	2	3
B_p	h	i	j

K: contracted dimension

It is important to align the contracted dimension to perform a valid compute

cycles (time)

Main Memory
A_p
B_C B_p

Compute

Read(A_c[0])
Read(A_p[0])
Read(B_c[0])
Read(B_p[0])

Read(B_c[1])
Read(B_p[1])

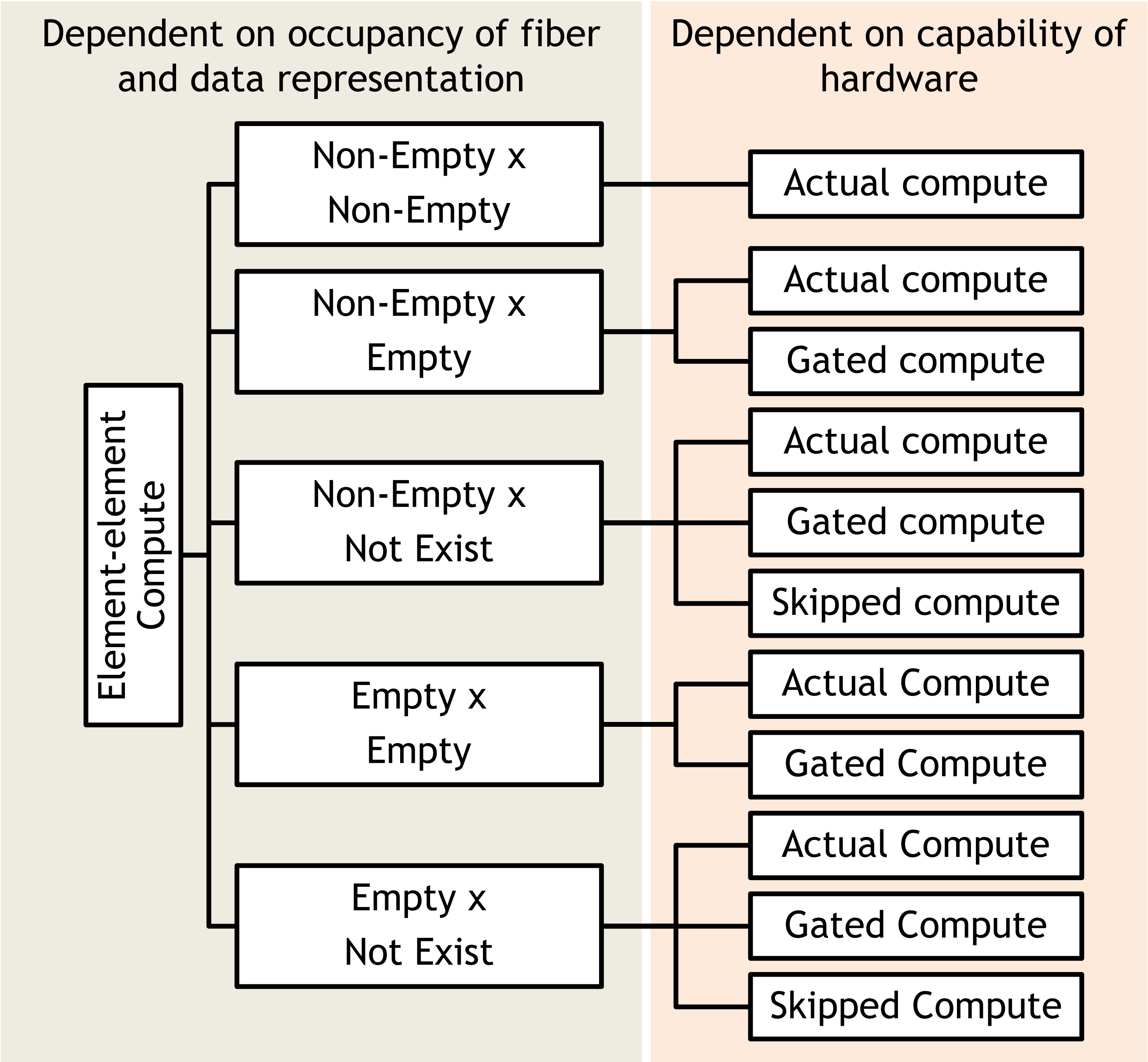
b.c. A[1] does not exist

Read(A_c[1])
Read(A_p[1])
Read(B_c[2])
Read(B_p[2])

Compute
(A_p[0], B_p[1])
c i

Compute
(A_p[1], B_p[2])
d j

Sparse Optimization Features Lead to Different Types of Computes

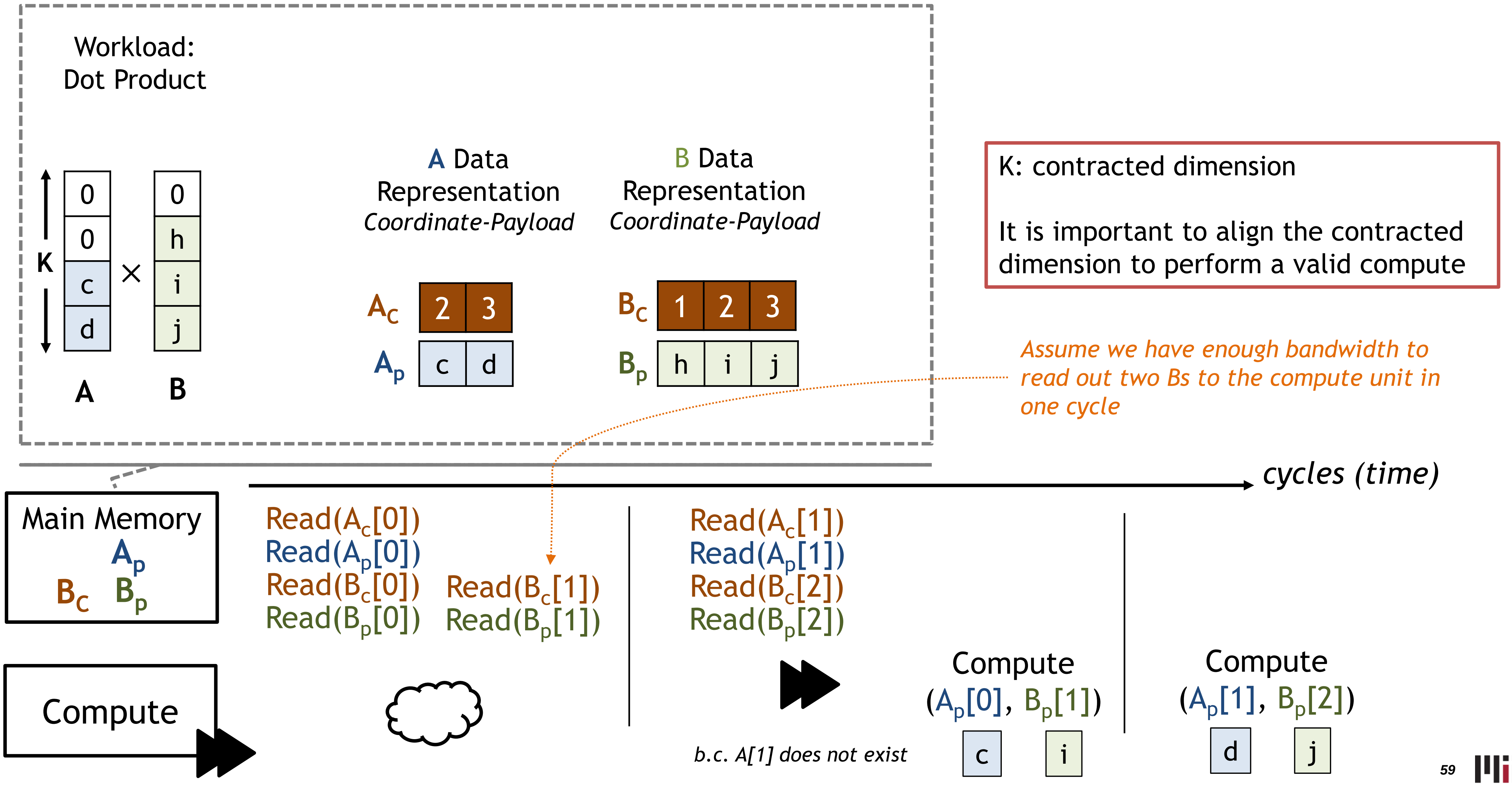


Gating:
Explicit energy saving of compute when one of the payloads of operand elements is empty (i.e., compute engine recognizing zero operands)

Skipping:
Explicit skipping over a compute when one of the payloads of operand elements does not exist (i.e., look-up based operand alignment)

Note: skipping cannot skip over empty elements

Skipped Compute Unit Working on Dot Product

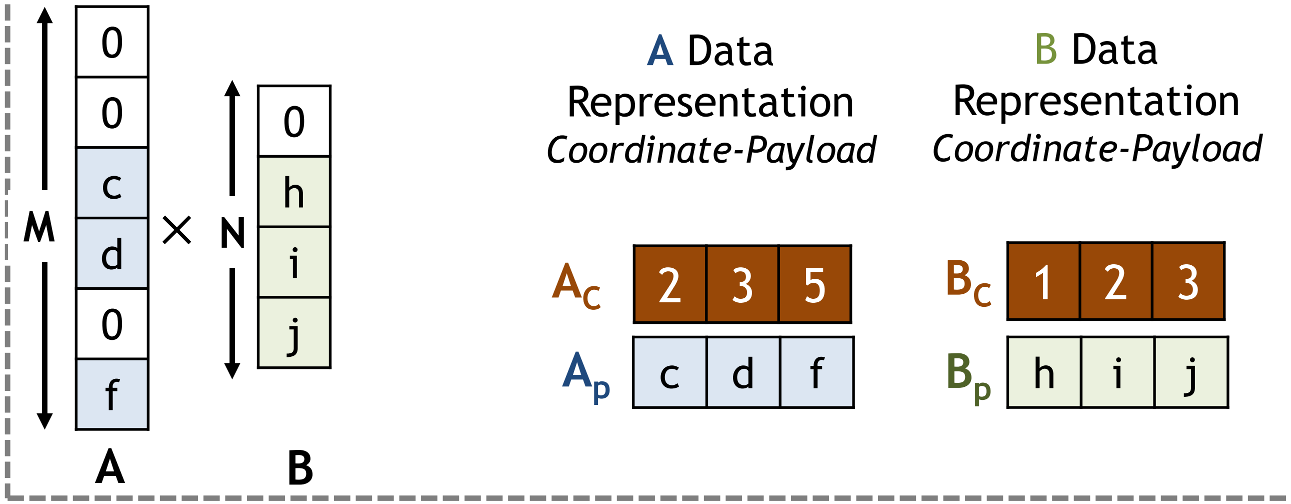


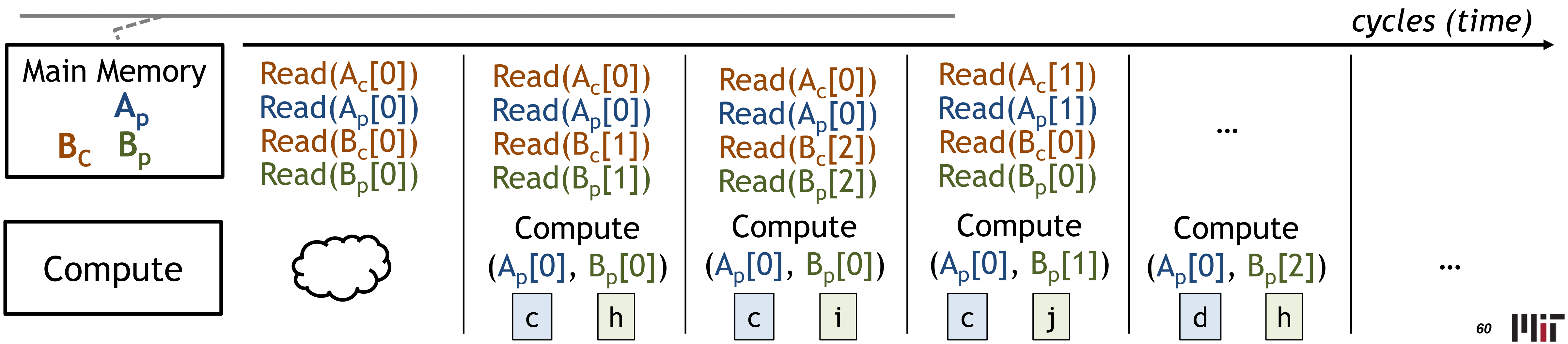
Baseline Compute Unit Working on Cross Product

Workload:
Cross Product

```
for m in [0:M)
 for n in [0:N)
   Z[m,n] = A[m]*B[n]
```

There is no contracted dimension in a cross product, no alignment needed





Interactions between Problem Spec and Opt. Features

Workload:
Cross Product

```
for m in [0:M)
 for n in [0:N)
   Z[m,n] = A[m]*B[n]
```

M	A	×	N	B
	0			0
	0			h
	c			i
	d			j
	0			
	f			

A Data Representation *Coordinate-Payload*

A_C	2	3	5
A_p	c	d	f

B Data Representation *Coordinate-Payload*

B_C	1	2	3
B_p	h	i	j

There is no contracted dimension in a cross product, no alignment needed

Gating/Skipping does not make a difference

cycles (time)

Main Memory
A_p
B_C B_p

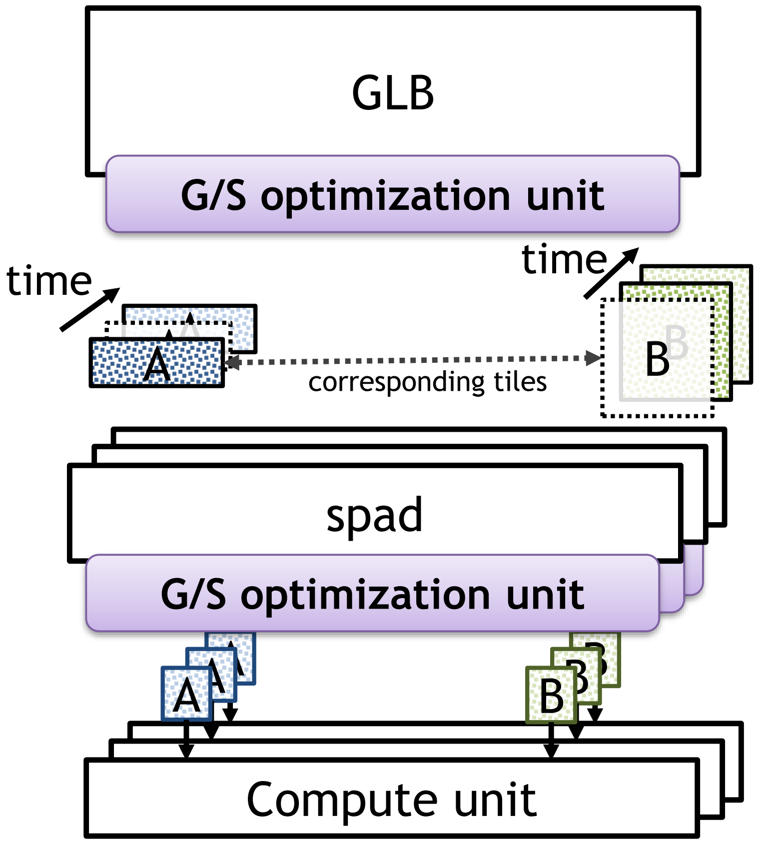
Compute

Read(A_c[0]) Read(A_p[0]) Read(B_c[0]) Read(B_p[0])	Read(A_c[0]) Read(A_p[0]) Read(B_c[1]) Read(B_p[1])	Read(A_c[0]) Read(A_p[0]) Read(B_c[2]) Read(B_p[2])	Read(A_c[1]) Read(A_p[1]) Read(B_c[0]) Read(B_p[0])	…	
	Compute (A_p[0], B_p[0]) c h	Compute (A_p[0], B_p[0]) c i	Compute (A_p[0], B_p[1]) c j	Compute (A_p[0], B_p[2]) d h	…

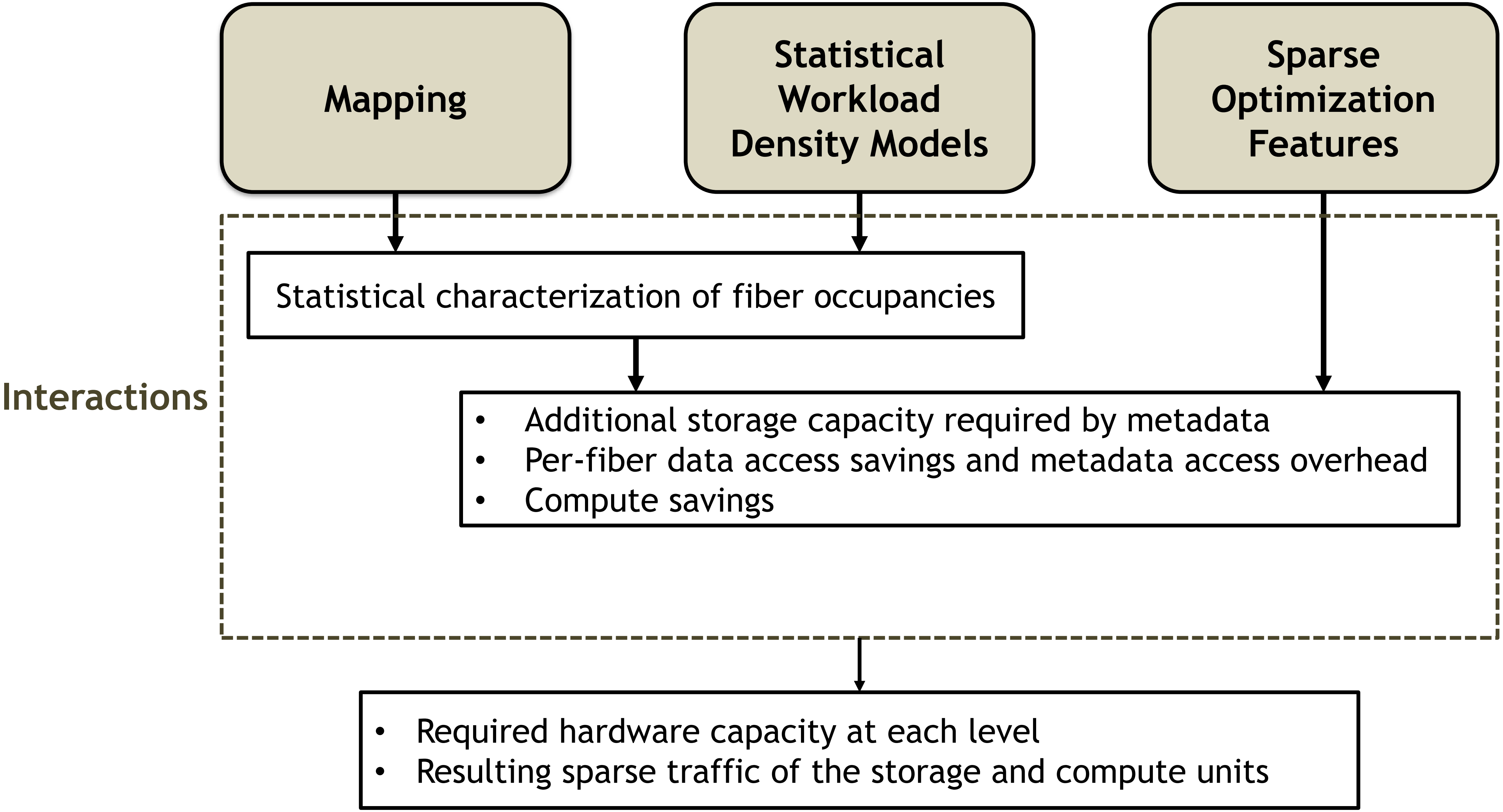
More Modeling Capabilities

- **Zero-Gating and Zero-Skipping at intermediate storage levels**
  - Propagation Impact to lower storage and compute levels
  - Choose gated/skipped tensor based on mapping
- **Multi-rank compression formats**
  - Interaction between compression formats and mapping
  - Compression with flattened ranks (important for deep neural network workloads)
  - Decompression at inner storage levels

More Realistic Multi-Level Architecture



Specifications and Their Interactions



Specifications and Their Interactions

Mapping

Statistical Workload Density Models

Sparse Optimization Features

Statistical characterization of fiber occupancies

- Additional storage capacity required by metadata
- Per-fiber data access savings and metadata access overhead
- Compute savings

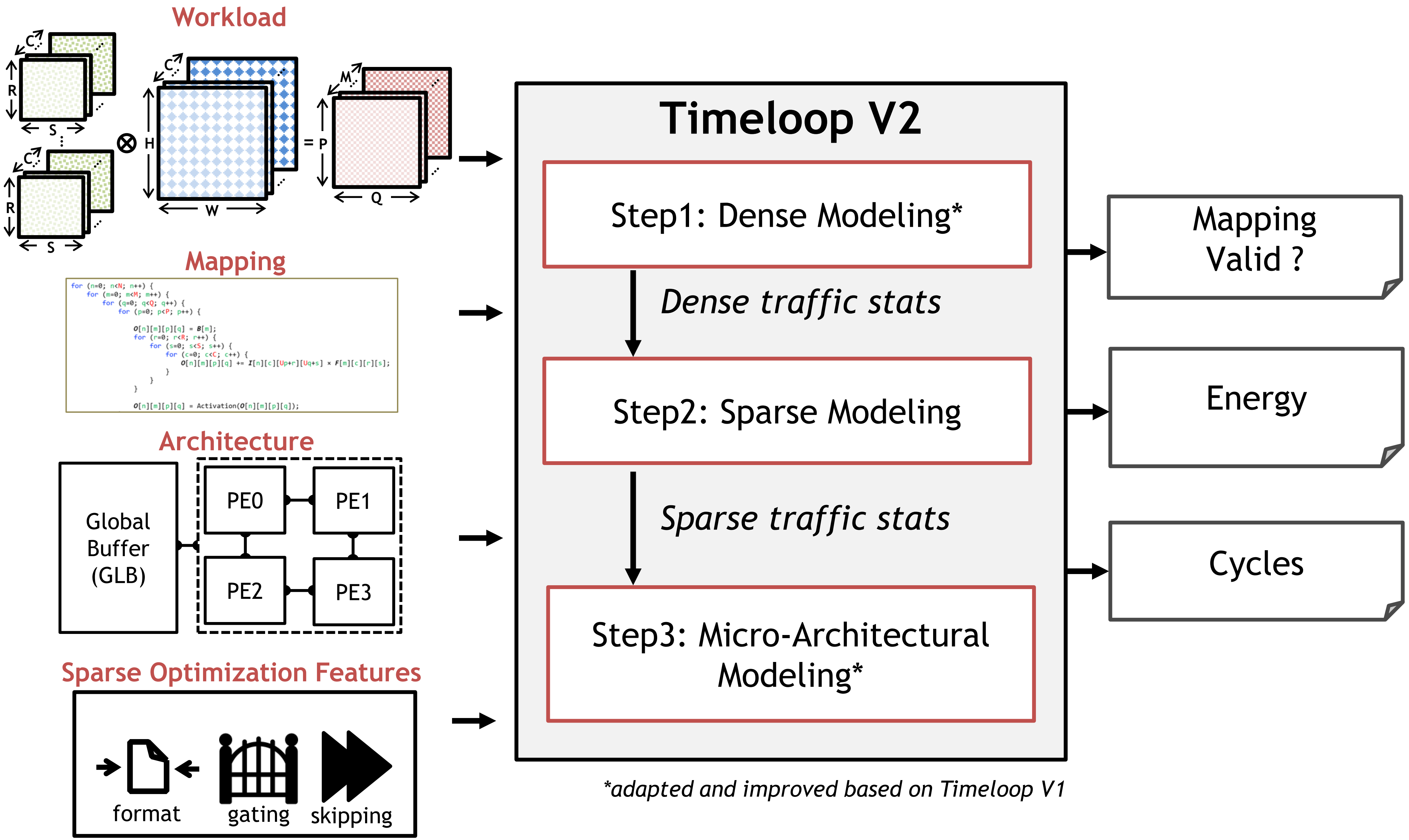
Decoupled dense-sparse analysis

Apply appropriate scaling based on dense traffic–fiber shape ratio

- Required hardware capacity at each level
- Resulting sparse traffic of the storage and compute units

Timeloop V2 (a.k.a. Sparseloop) Infrastructure

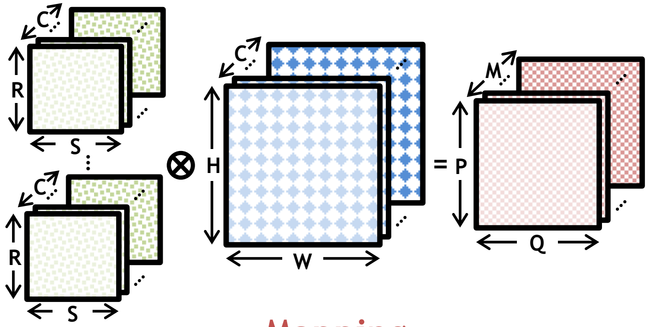
Timeloop V2



**adapted and improved based on Timeloop V1*

Timeloop V2 Inputs

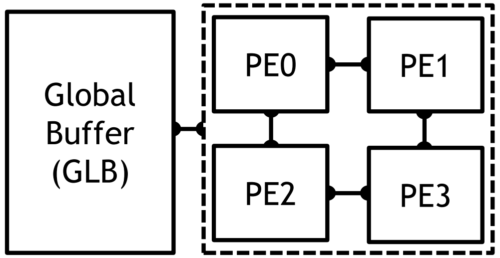
Workload



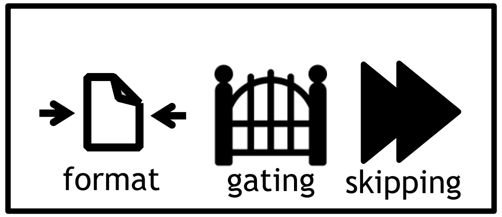
Mapping

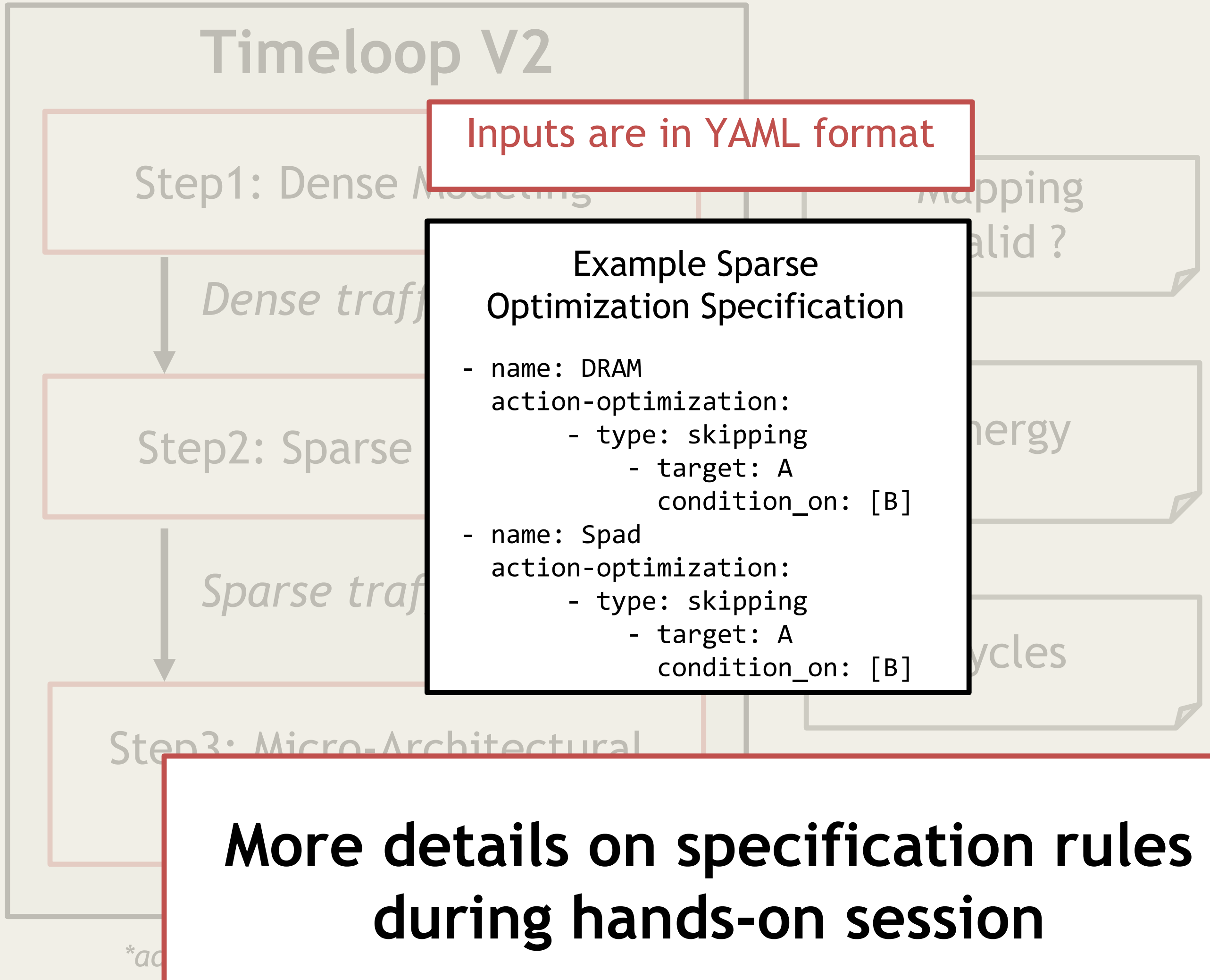
```
for (n=0; n<N; n++) {
    for (m=0; m<M; m++) {
        for (q=0; q<Q; q++) {
            for (p=0; p<P; p++) {

                O[n][m][p][q] = B[m];
                for (r=0; r<R; r++) {
                    for (s=0; s<S; s++) {
                        for (c=0; c<C; c++) {
                            O[n][m][p][q] += I[n][c][Up+r][Uq+s] × F[m][c][r][s];
                        }
                    }
                }

                O[n][m][p][q] = Activation(O[n][m][p][q]);
```

Architecture



Sparse Optimization Features





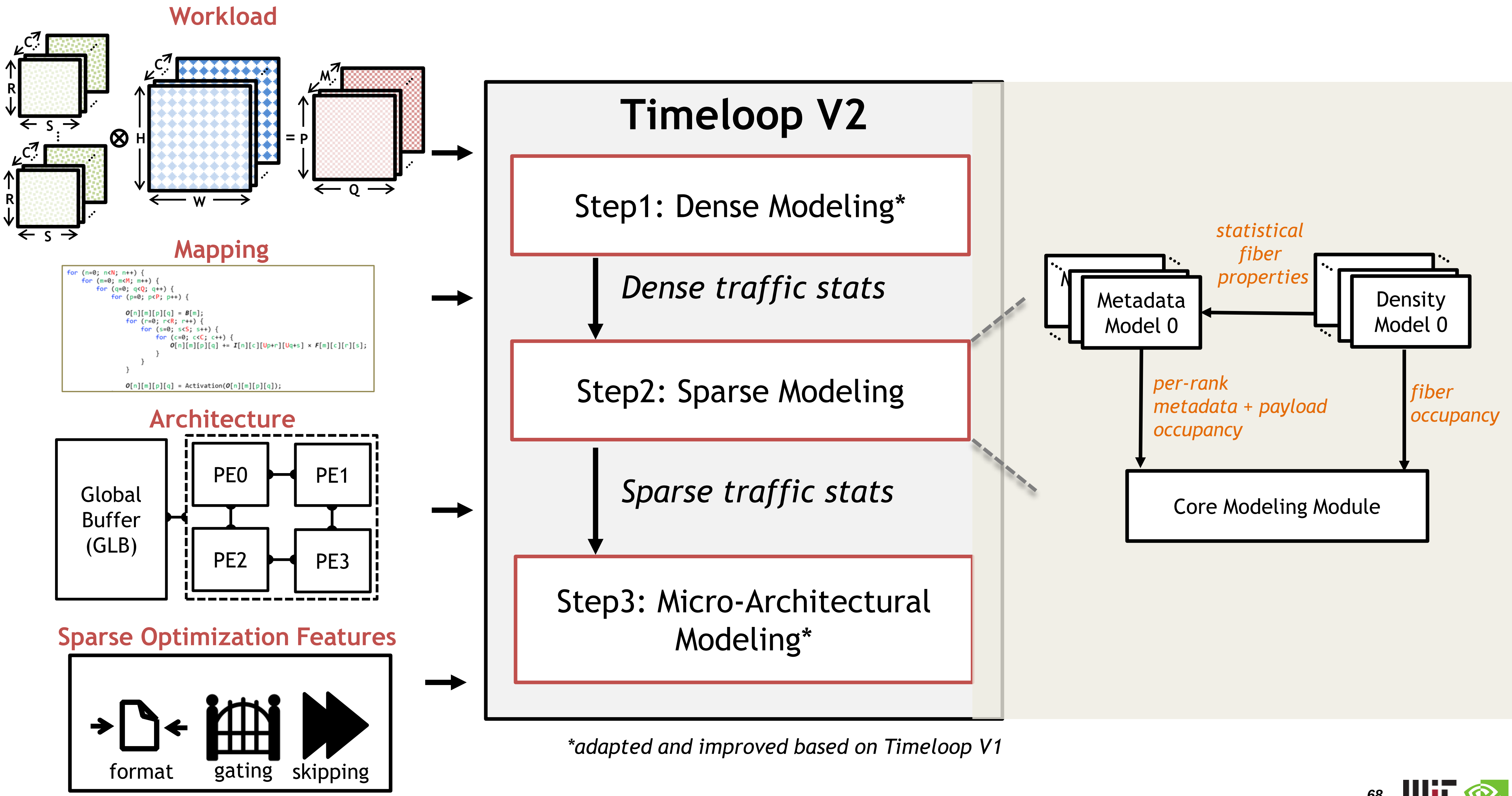
Inputs are in YAML format

Example Sparse Optimization Specification

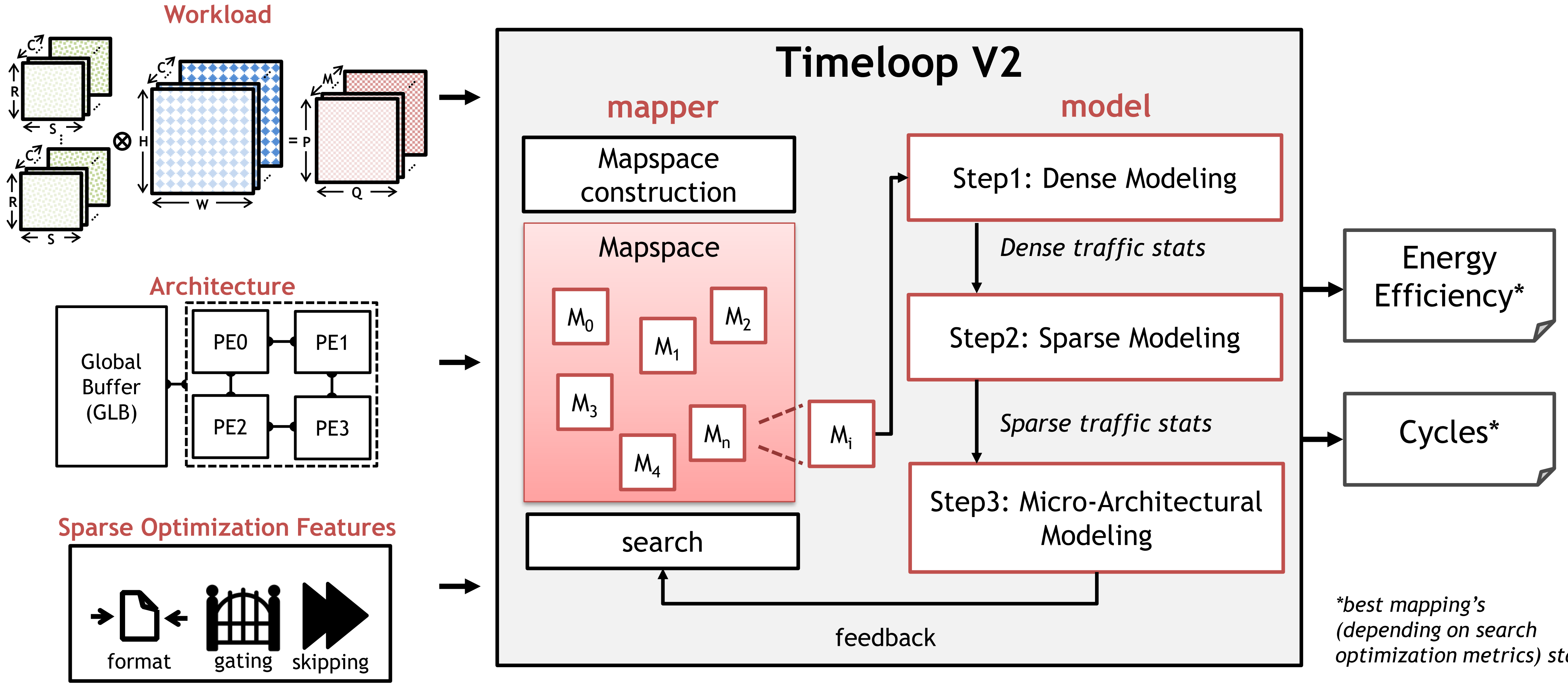
```
- name: DRAM
  action-optimization:
        - type: skipping
            - target: A
              condition_on: [B]
- name: Spad
  action-optimization:
        - type: skipping
            - target: A
              condition_on: [B]
```

More details on specification rules during hands-on session

Modularized Density and Format Models



Timeloop V2 Mapspace Exploration

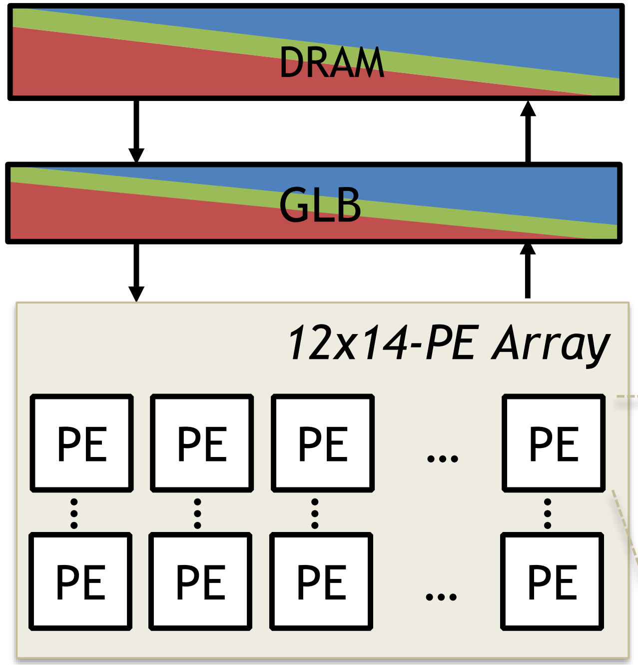


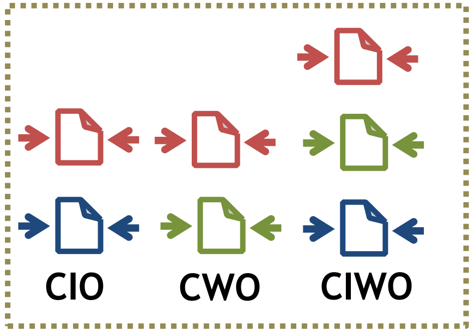
*best mapping's (depending on search optimization metrics) stats

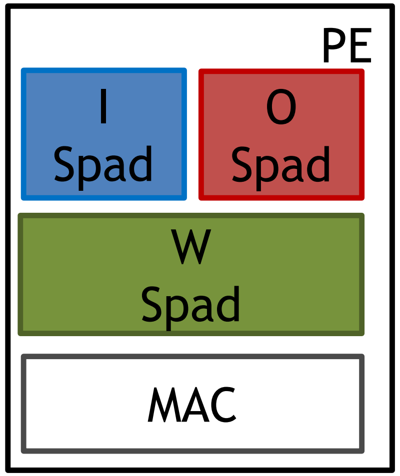
Case Studies

Explore different sparse optimization features

High-Level Architecture Setup



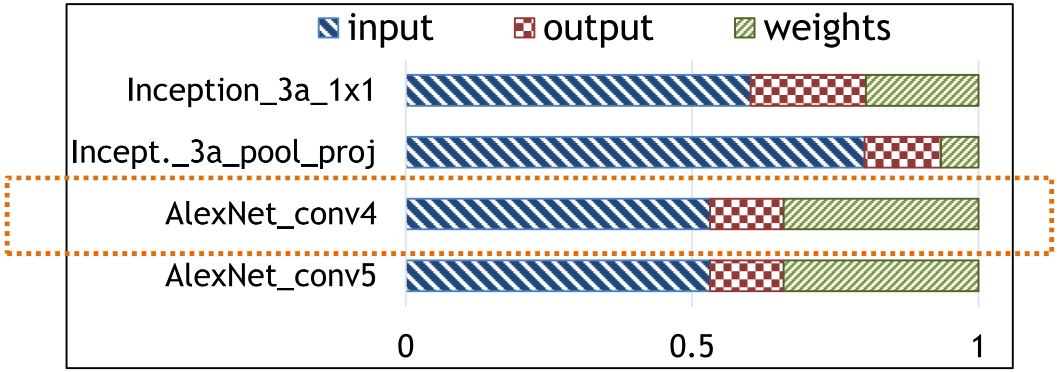




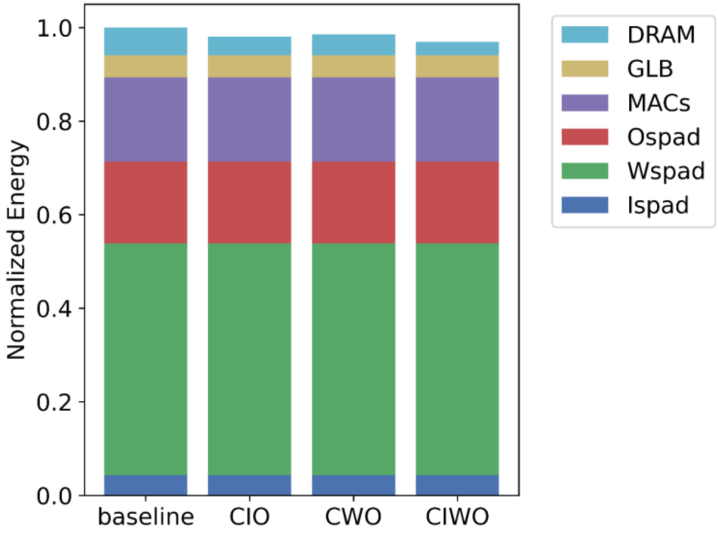
What are some important factors that define the impact of compressed data representation format?

Uncompressed Traffic Breakdown vs. Compression Savings

Uncompressed DRAM Traffic Breakdown



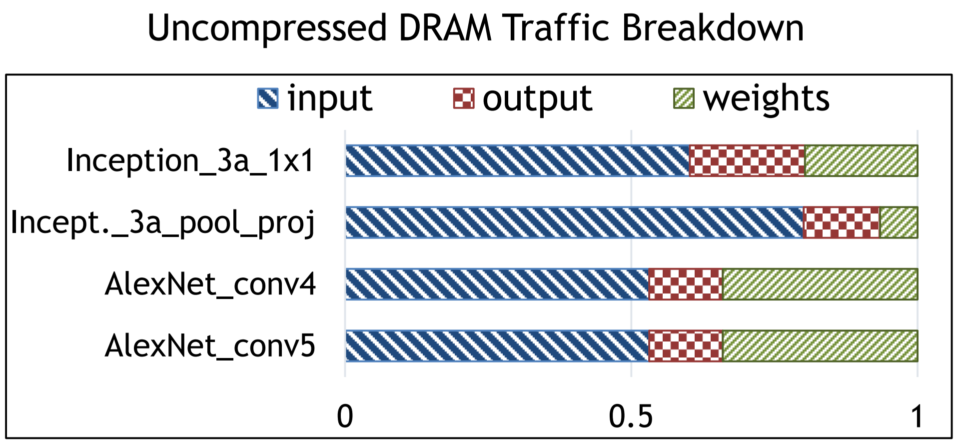
AlexNet Conv4



The tensor that dominates uncompressed traffic introduces more savings when compressed

Is that true? No

Tensor Densities Play an Important Role

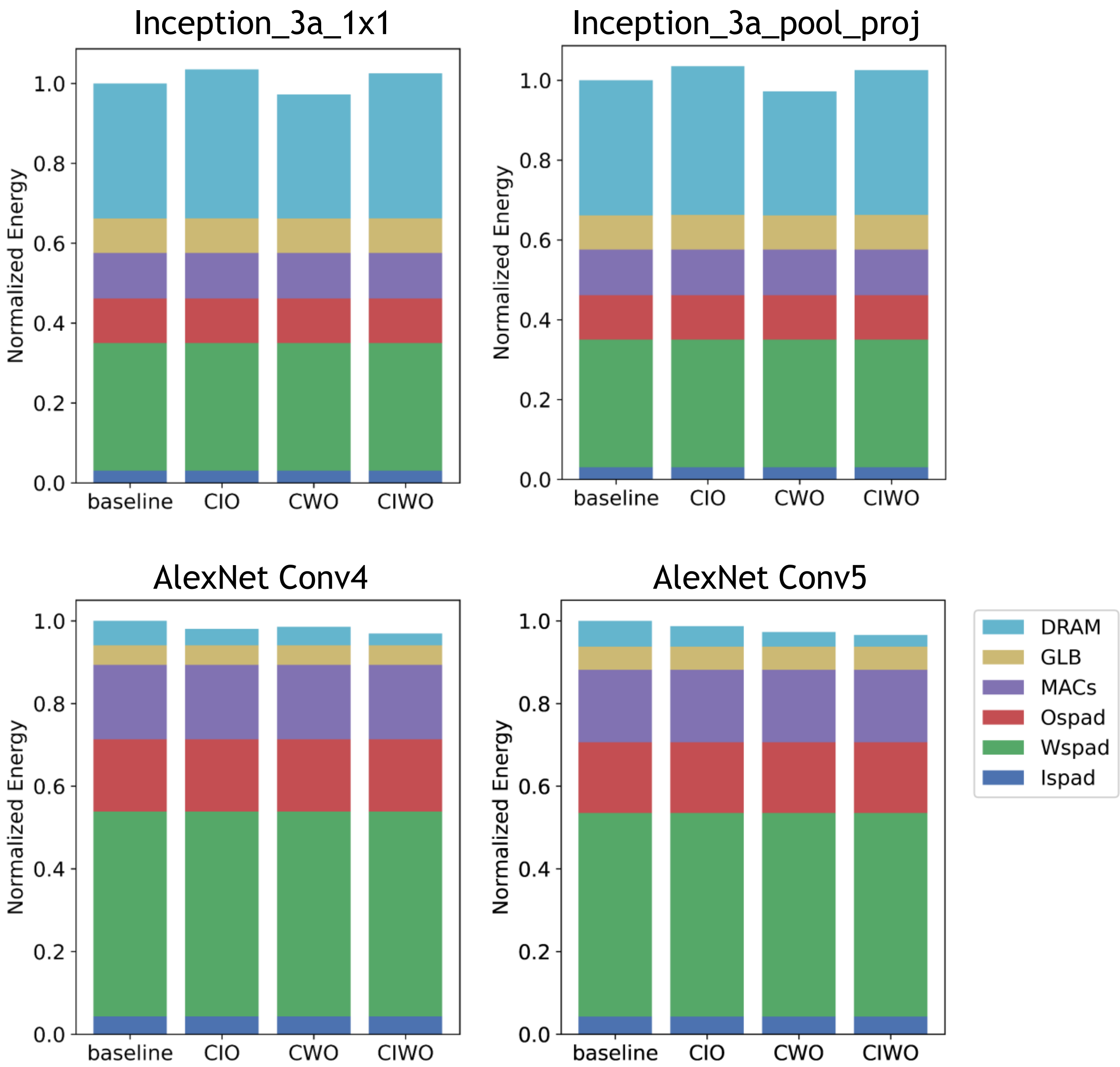


The tensor that dominates uncompressed traffic introduces more savings when compressed

Is that true? No

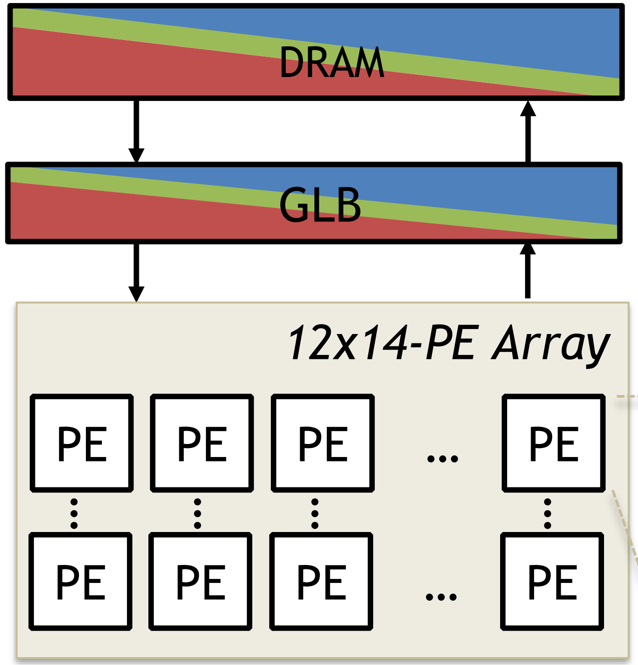
Layer Densities

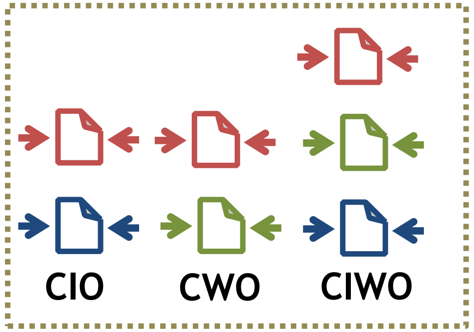
Layer #	Inputs	Outputs	Weights
Inception_3a_1x1	0.71	0.66	0.37
Incept._3a_pool_proj	0.96	0.46	0.46
Alexnet_conv4	0.39	0.43	0.37
Alexnet_conv5	0.43	0.16	0.37



Explore different sparse optimization features

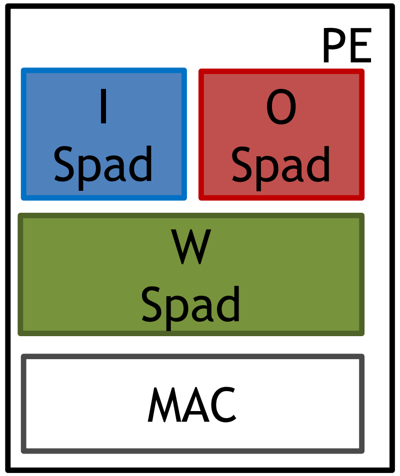
High-Level Architecture Setup

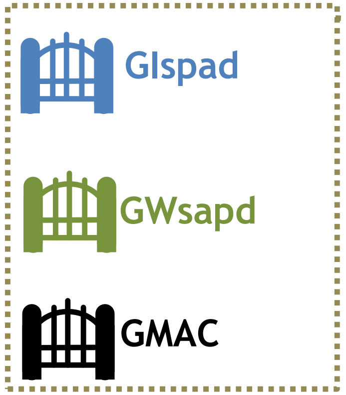




What are some important factors that define the impact of compressed data representation format?

- *Uncompressed traffic breakdown*
- *Tensor density*





What are some important factors that define the impact of gating on-chip?

Density vs. Gating Savings

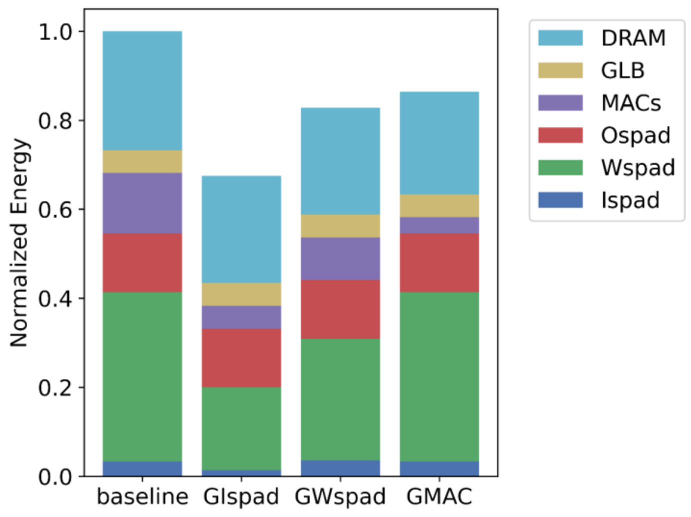
Layer Densities

Layer #	Inputs	Outputs	Weights
Inception_3a_1x1	0.71	0.66	0.37
Incept._3a_pool_proj	0.96	0.46	0.46
Alexnet_conv4	0.39	0.43	0.37
Alexnet_conv5	0.43	0.16	0.37

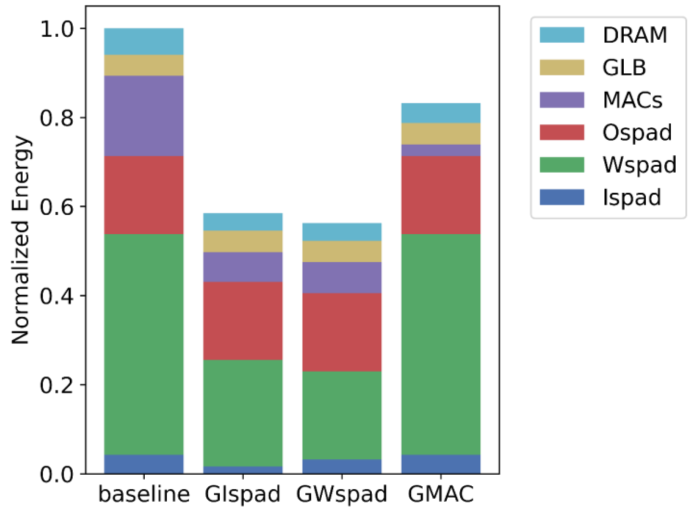
The tensor that has lower density should be the conditioned on tensor, i.e., it should have associated with metadata and allows the other tensor to be gated

Is that true? No

Inception_3a_1x1



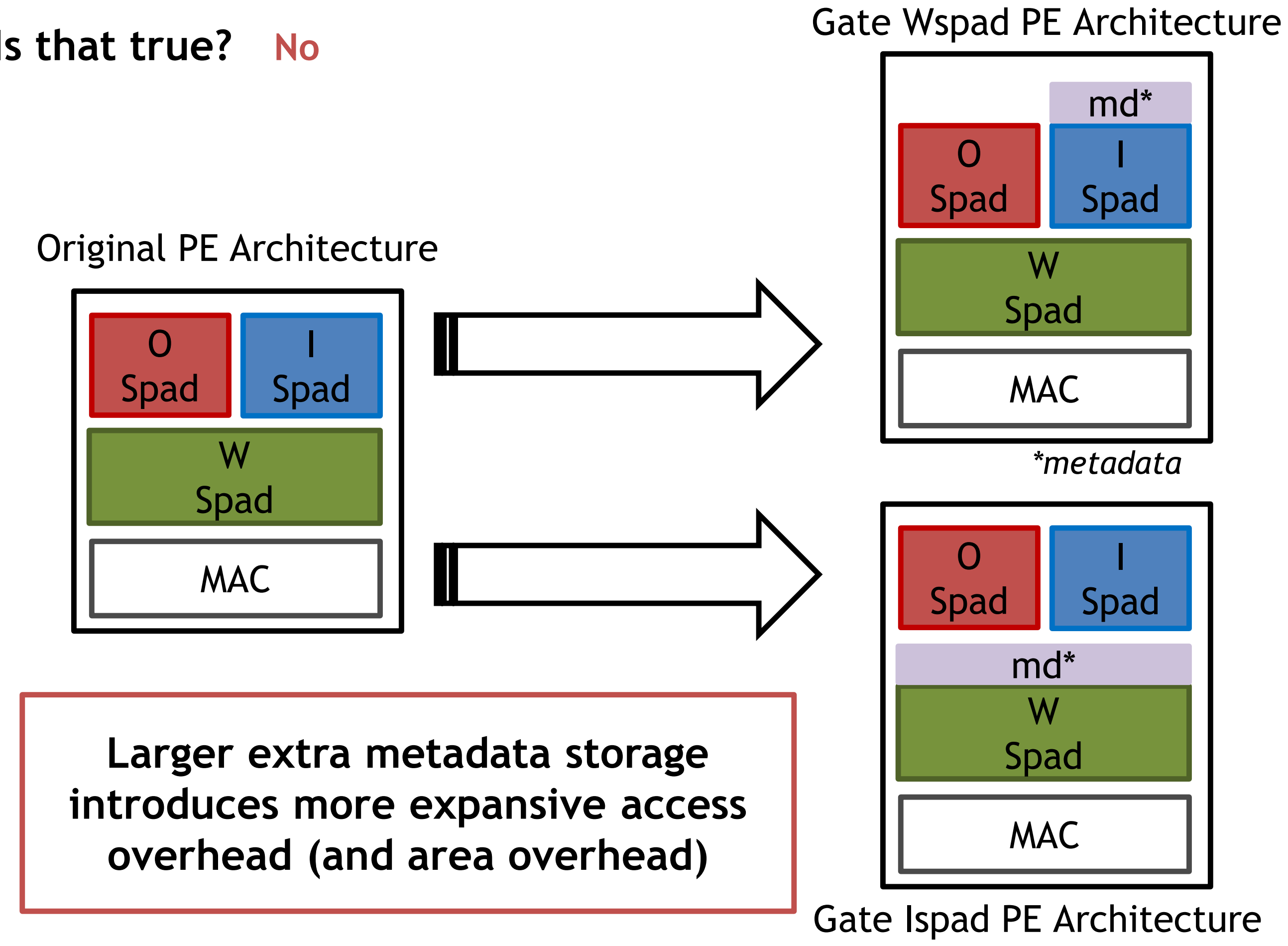
AlexNet Conv4



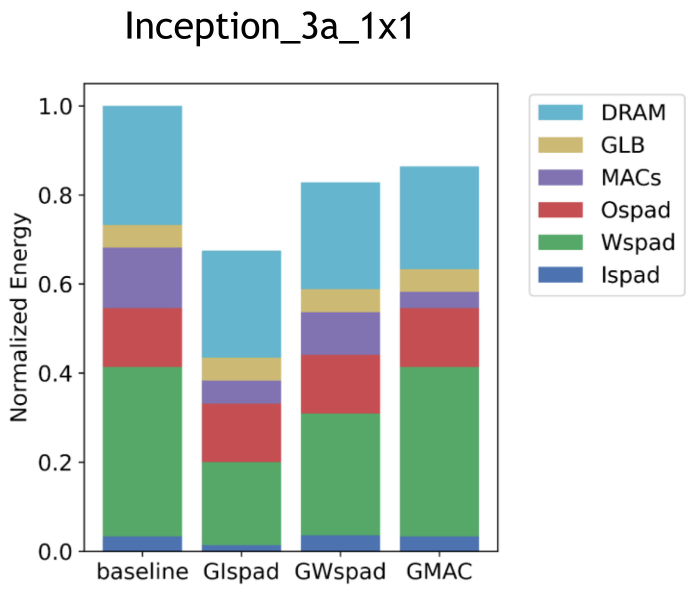
Hardware Attirbutes Plays an Important Role

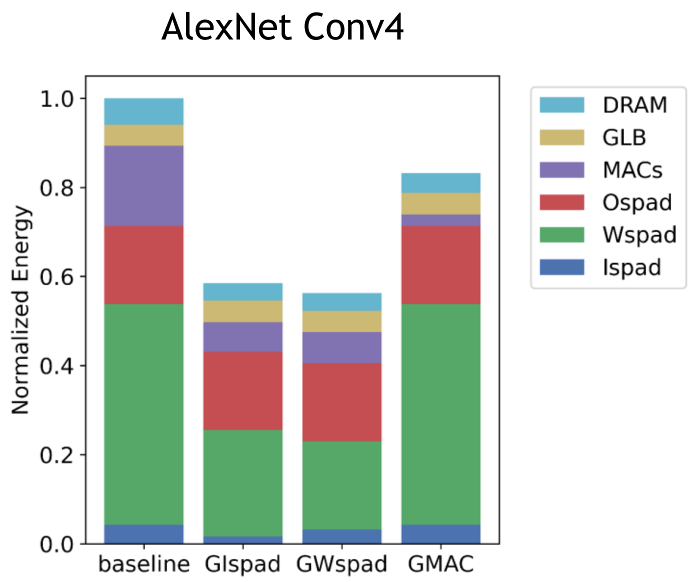
The tensor that has lower density should be the conditioned on tensor, i.e., it should have associated with metadata and allows the other tensor to be gated

Is that true? No



Larger extra metadata storage introduces more expansive access overhead (and area overhead)



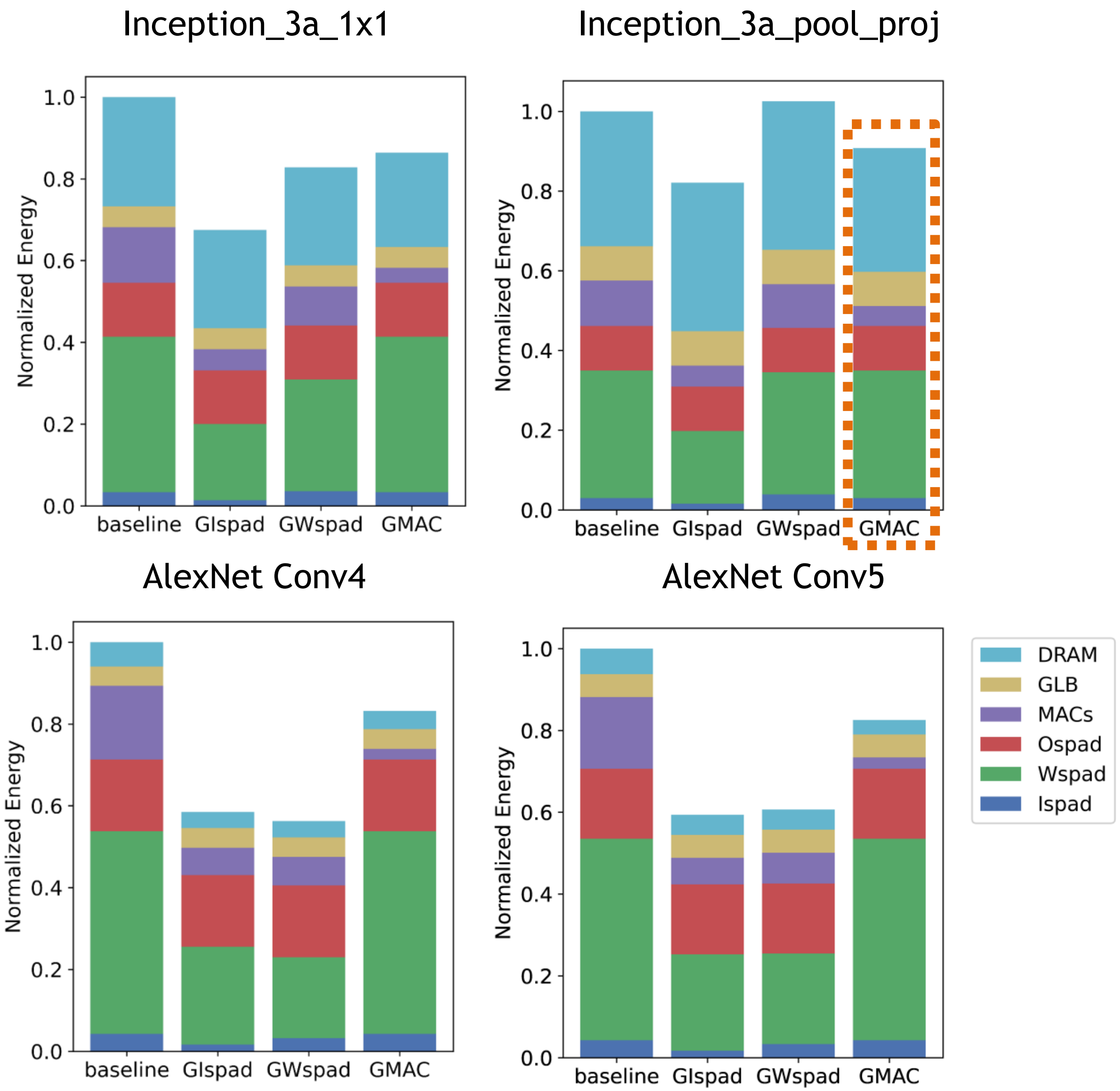


More Examples

Layer Densities

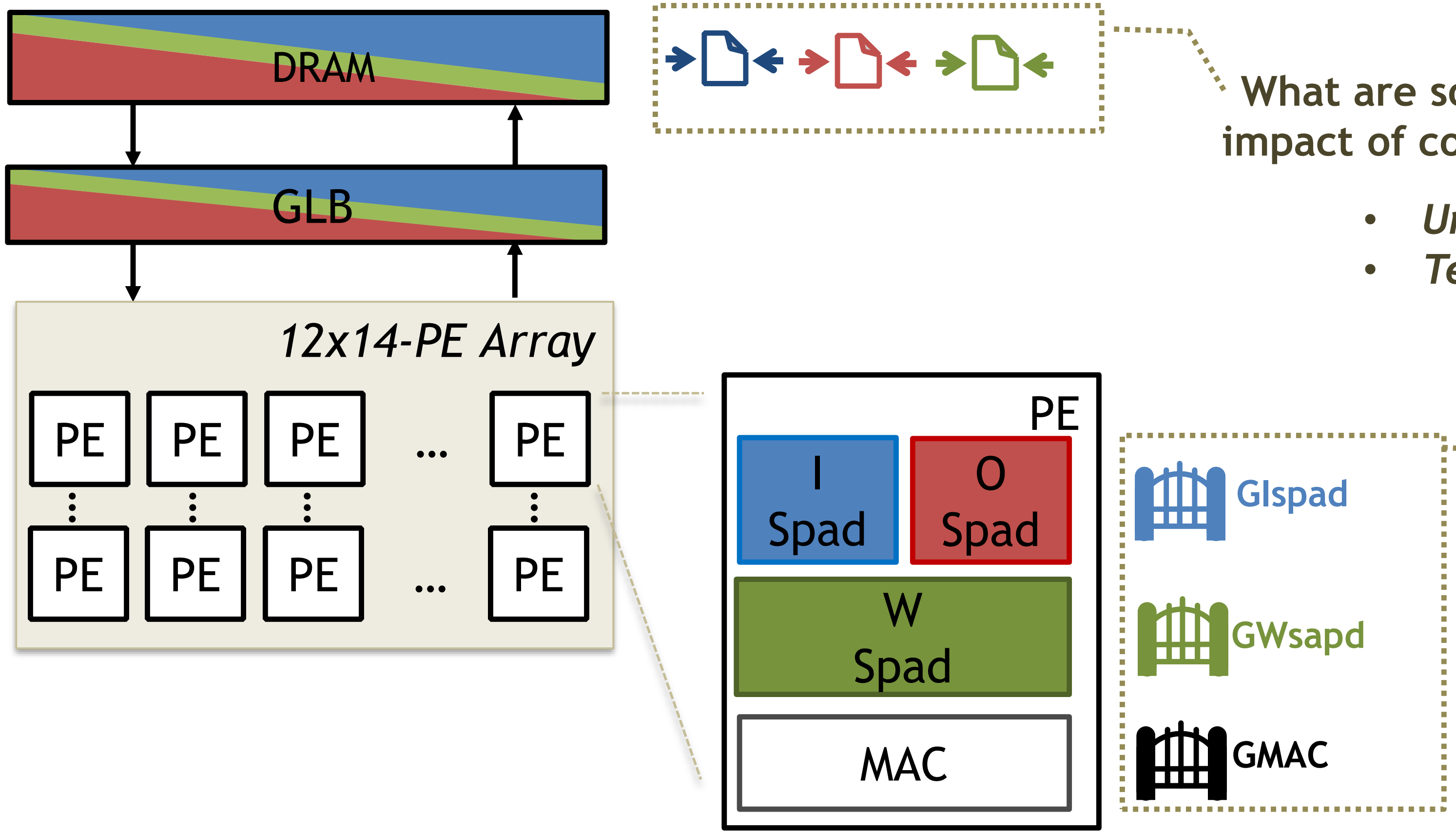
Layer #	Inputs	Outputs	Weights
Inception_3a_1x1	0.71	0.66	0.37
Incept._3a_pool_proj	0.96	0.46	0.46
Alexnet_conv4	0.39	0.43	0.37
Alexnet_conv5	0.43	0.16	0.37

Gate compute only could introduce better energy efficiency (and simpler hardware)



Explore different sparse optimization features

High-Level Architecture Setup



What are some important factors that define the impact of compressed data representation format?

- *Uncompressed traffic breakdown*
- *Tensor density*

What are some important factors that define the impact of gating on-chip?

- *Uncompressed traffic*
- *Tensor density*
- *Hardware attributes*

Sparse Tensor Accelerator Modeling Summary

- Methodology
  - Specifications
    - Mapping
    - Statistical workload density models
    - Sparse optimization features
  - Systematic analysis of the interactions between different specifications
  - Modularized modeling process that decouples dense traffic modeling and sparse optimization impact modeling
- Timeloop V2 (a.k.a. Sparseloop) Infrastructure
  - Implements the proposed methodology based on Timeloop V1
  - Modularized to allow data representation format and density model plug-ins
- Validation and case studies
  - Validation on Eyeriss V1 and SCNN
  - Exploration of various combinations of sparse optimization features

Sparse Tensor Accelerators: Abstraction and Modeling

Background Lecture Part 2

Joel Emer
Angshuman Parashar
Vivienne Sze
Po-An Tsai
Nellie Wu

ISCA Tutorial

June 2021

NVIDIA

Massachusetts Institute of Technology